THE DEPUTY SECRETARY OF DEFENSE

WASHINGTON, D.C. 20301

JAN 2 1981

MEMORANDUM FOR SECRETARIES OF THE MILITARY DEPARTMENTS
CHAIRMAN, JOINT CHIEFS OF STAFF
DIRECTOR, DEFENSE ADVANCED RESEARCH PROJECTS AGENCY
DIRECTOR, DEFENSE COMMUNICATIONS AGENCY
DIRECTOR, DEFENSE INTELLIGENCE AGENCY
DIRECTOR, DEFENSE INVESTIGATIVE SERVICE
DIRECTOR, DEFENSE LOGISTICS AGENCY
DIRECTOR, DEFENSE MAPPING AGENCY
DIRECTOR, DEFENSE NUCLEAR AGENCY
DIRECTOR, NATIONAL SECURITY AGENCY
DIRECTOR, WWMCC SYSTEM ENGINEERING

SUBJECT: DOD Computer Security Evaluation Center

Although your comments in response to Dr. Dinneen's memorandum of November 13 indicate some concern about working relationships within the proposed Evaluation Center, there is no disagreement or doubt regarding the need. Therefore, the proposal made by the Director, National Security Agency to establish a Project Management Office is approved. Effective January 1, 1981, the Director, National Security Agency is assigned the responsibility for Computer Security Evaluation for the Department of Defense.

Please provide the name of your representative for computer security matters to ASD(C^3I). The individual chosen for this task should be empowered to work in your behalf to develop and coordinate the charter and implementing directives for the Center. I expect this working group to identify necessary personnel and fiscal resources.

W. Graham Claytor

W. Graham Claytor, Jr.

cc: ASD(C^3I)
ASD(Comptroller)
DUSD(Policy Review)

Figure 1-1

34344

assessment of the progress to date in achieving widespread availability of trusted computer systems.

The computer manufacturers are making substantial progress in improving the integrity of their products, as can be seen by a review of section 3 of this report. Most of the incentive for this comes from a strong need to build more reliable and easily maintainable products coupled with a significant increase in the computer science understanding of how to produce more reliable hardware and software. This trend was well established before the efforts of the Initiative and can be expected to continue at an accelerating pace. But the existence of an organized effort on the part of the government to understand the integrity measures of industry developed computer products will have a strong influence on the evolution of the industry's integrity improvement measures.

If the government can establish consistent evaluation criteria, the efforts of the Initiative to date have shown that the industry will evolve their systems in accordance with those criteria and the government can then expect to be able to purchase high integrity computer products in the same manner they purchase standard ADP systems today, without the high additional costs of special purpose development and maintenance. This is the philosophy being pursued by the Initiative, to influence the evolution of highly reliable commercial products to enable their use in sensitive information handling applications and to obtain sufficient understanding of the integrity of individual products to determine suitable environments for their use.

This report is organized in the following manner. The remainder of this section summarizes the major activities of the Initiative since June 1978. Section 2 gives background on the general nature of the computer security problem and some technical details helpful in understanding the trusted system evaluation process. Section 3 describes the current status of the Initiative, including: (1) a description of the Evaluated Products List concept, (2) a description of the Trusted Computing Base (TCB) concept, (3) current draft evaluation criteria, (4) a proposed evaluation process, and (5) the status of current Initiative evaluation efforts. Section 4 describes ongoing R&D, plans, and industry implications in the areas of trusted operating systems, trusted applications, and verification technology.

1.1 COMPUTER SECURITY INITIATIVE ACTIVITIES

Figure 1-2 illustrates the overall activities of the Initiative. There are three main efforts being pursued in parallel. The eventual outcome of this work is the

COMPUTER SECURITY INITIATIVE

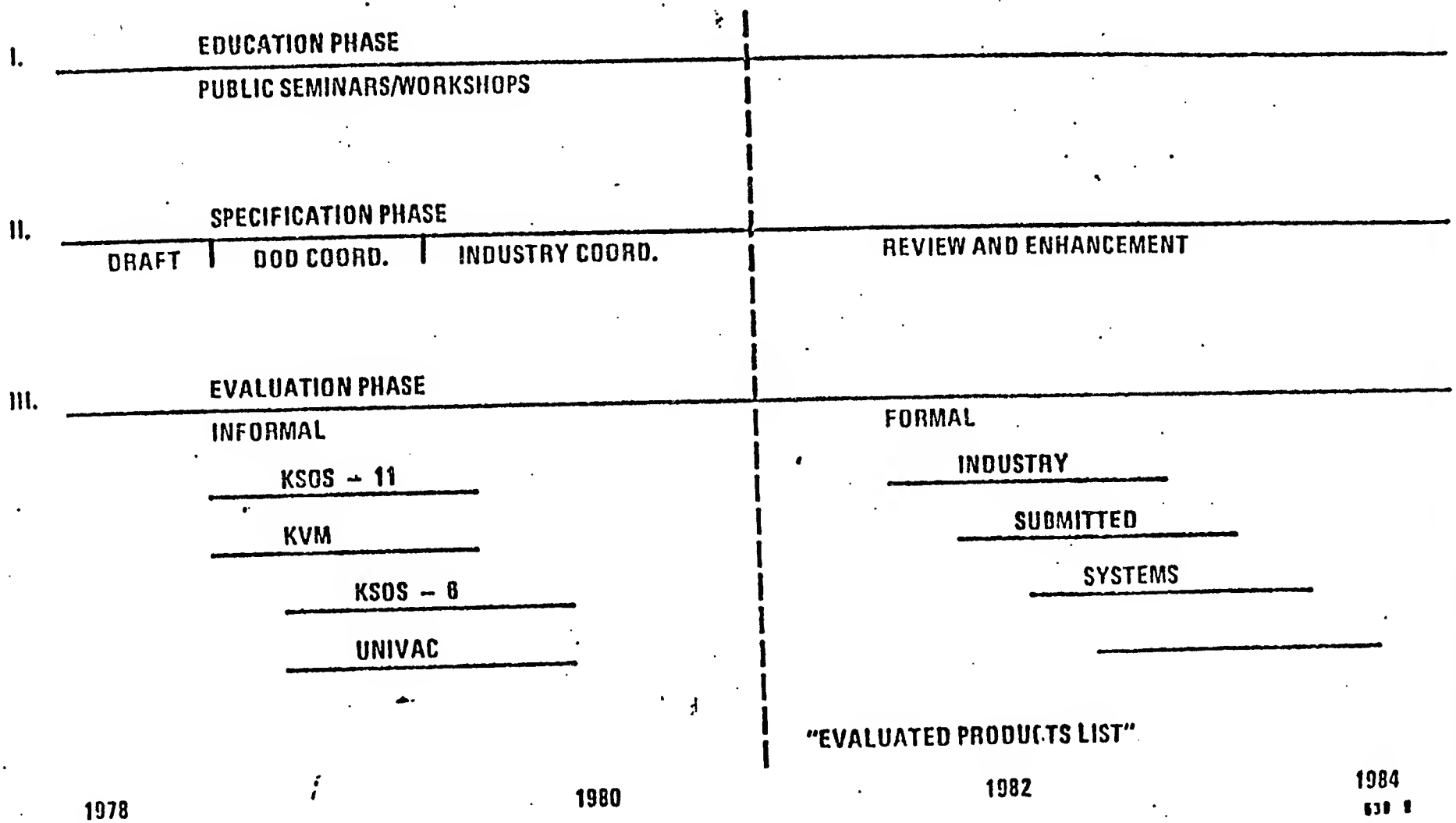

Figure 1-2

Approved For Release 2007/06/01 : CIA-RDP83M00914R001800060009-2

establishment of a consistent and systematic means of evaluating the integrity of industry and government developed computer systems. This outcome will be accomplished when the Initiative has reached the Formal Evaluation of industry developed systems represented in the lower right of the figure. Before this can happen, the evaluation process must be formalized, criteria for evaluation established and an Executive Agent identified to carry out the evaluations. The vertical dotted line represents the accomplishment of this formalization. Prior to this, in the Specification Phase (Part II on the figure), draft evaluation criteria and specifications for a "Trusted Computing Base" (TCB) are being developed (see section 3 of this report). These draft documents are being distributed for comment to the DoD through the Consortium and to industry through the Education Phase efforts described below. In order to ensure that the draft criteria and specifications are realistic and feasible, the Initiative has been conducting, at the invitation of various computer manufacturers, evaluations of several potential industry trusted systems. (Section 3.5 describes present efforts). These informal evaluations are performed by members of the Consortium, governed by OSD General Council approved legal limitations and non-disclosure agreements. They are conducted as mutually beneficial technical discussions with the manufacturers and are serving a vital function in illustrating the feasibility of such an evaluation process and the industry's strong interest and willingness to participate.

The other major part of the Initiative's efforts as represented on figure 1-2 is the Education Phase. The goal in this effort is two fold: (1) to transfer technology to the computer manufacturers on how to develop trusted computer systems and (2) to identify to the general computer user community that trusted computers can be built and successfully employed in a wide variety of applications. The principle method of accomplishing these goals is through public seminars. Three such seminars have been held at the National Bureau of Standards in July 1979, January 1980, and November 1980. These seminars were attended by 300-350 people representing all the major computer manufacturers, over 50 computer system user organizations and over 25 Federal and State organizations. The seminars have generated a great deal of interest in the development of trusted computer systems. In addition frequent participation in national level conferences such as the National Computer Conference (1979 and 1980) have helped to establish the viability of the trusted computer concept.

There are three major efforts in the DoD computer security R&D program. The first is the development and demonstration of trusted operating systems. Included in these efforts are the Kernelized Secure Operating System (KSOS), which went into initial test site evaluation during the fall of 1980, and the Kernelized VM/370 System (KVM/370), which will be installed at two test sites by the first quarter of 1981. Also included in this activity is the hardware and security kernel development efforts on the Honeywell Secure Communications Processor (SCOMP). All of these efforts began as DARPA programs with joint funding from many sources. Through the efforts of the Initiative, arrangements have been made, starting in Oct 1980, for the Navy to assume technical and contractual responsibility for the KSOS and SCOMP efforts and for the Air Force to assume similar responsibility for the KVM/370 effort. These efforts are essential for the demonstration of trusted computer systems to the DoD and also as examples to the manufacturers as incentives to produce similar systems.

The second major R&D activity is the development of applications of trusted computer systems. These include the various guard-related information sanitization efforts (e.g. ACCAT GUARD, FORSCOM GUARD), trusted front-end systems (e.g. COINS Trusted TAS, DCA COS-NFE), trusted message system activities (e.g. DARCOM Message Privacy Experiments), and a recently-started effort in trusted data base management systems.

The third R&D thrust is the establishment of a verification technology program to advance the state of the art in trusted system specification and verification. The first phase of this program (FY80-FY83) includes major competitive procurement activities to broaden our experience in using current program verification technologies. This effort is being undertaken to better understand the strengths and weaknesses of these systems in order to be able to better specify our requirements for future improved systems which will be developed in the second phase of the program (FY83-FY86). The Air Force has a major effort in this area beginning in FY81. The Navy is initiating an R&D effort to integrate several existing technologies into a package for the specification and verification of applications like the various Guard systems now under development.

A significant factor in the progress of the DoD R&D activities in the past year has been the actions taken in response to recommendations of the Defense Oversight Committee's Report, "Employing Computer Security Technology to Combat Computer Fraud." The Committee's report recommended that the Services establish long-term programs in computer security R&D and that specific sums be allocated by each service in FY79, 80 and 81 while these long term

programs are being established. The FY80 funds recommended by the Committee were provided by March 1980 and have been instrumental in keeping ongoing efforts underway and providing the resources needed to establish the new application and verification technology development efforts.

SECTION 2

BACKGROUND

The Defense Science Board Task Force on Computer Security described the nature of the computer security problem in a report entitled "Security Controls for Computer Systems" dated February 1970 [WARE70]. That description remains valid today and is reprinted here in part to set the context for this report.

2.1 NATURE OF THE PROBLEM

2.1.1 The Security Problem

"The wide use of computers in military and defense installations has long necessitated the application of security rules and regulations. A basic principle underlying the security of computer systems has traditionally been that of isolation--simply removing the entire system to a physical environment in which penetrability is acceptably minimized. The increasing use of systems in which some equipment components, such as user access terminals, are widely spread geographically has introduced new complexities and issues. These problems are not amenable to solution through the elementary safeguard of physical isolation.

"In one sense, the expanded problems of security provoked by resource-sharing systems might be viewed as the price one pays for the advantages these systems have to offer. However, viewing the question from the aspect of such a simplistic tradeoff obscures more fundamental issues. First, the security problem is not unique to any one type of computer system or configuration; it applies across the spectrum of computational technology. While the present paper frames the discussions in terms of time-sharing or multiprogramming, we are really dealing not with system configurations, but with security; today's computational technology has served as catalyst for focusing attention on the problem of protecting classified information resident in computer systems.

"Secondly, resource-sharing systems, where the problems of security are admittedly most acute at present, must be designed to protect each user from interference by another user or by the system itself, and must provide some sort of "privacy" protection to users who wish to preserve the integrity of their data and their programs. Thus, designers and manufacturers of resource-sharing systems are concerned with the fundamental problem of protecting information. In

protecting classified information, there are differences of degree, and there are new surface problems, but the basic issues are generally equivalent. The solutions the manufacturer designs into the hardware and software must be augmented and refined to provide the additional level of protection demanded of machines functioning in a security environment.

2.1.2 Types of Computer Systems

"There are several ways in which a computer system can be physically and operationally organized to serve its users. The security controls will depend on the configuration and the sensitivity of data processed in the system. The following discussion presents two ways of viewing the physical and operational configurations.

2.1.2.1 Equipment Arrangement and Disposition

"The organization of the central processing facilities for batch or for time-shared processing, and the arrangement of access capabilities for local or for remote interaction are depicted in figure 2-1. Simple batch processing is the historical and still prevalent mode of operation, wherein a number of jobs or transactions are grouped and processed as a unit. The batches are usually manually organized, and for the most part each individual job is processed to completion in the order in which it was received by the machine. An important characteristic of such single-queue, batched, run-to-completion systems which do not have an integrated file management system for non-demountable, on-line memory media is that the system need have no "management awareness" from job to job. Sensitive materials can be erased or removed from the computer quickly and relatively cheaply, and mass memory media containing sensitive information can be physically separated from the system and secured for protection. This characteristic explains why solution to the problem we are treating has not been as urgent in the past.

"In multiprogramming, on the other hand, the jobs are organized and processed by the system according to algorithms designed to maximize the efficiency of the total system in handling the complete set of transactions. In local-access systems, all elements are physically located within the computer central facility; in remote-access systems, some units are geographically distant from the central processor and connected to it by communication lines.

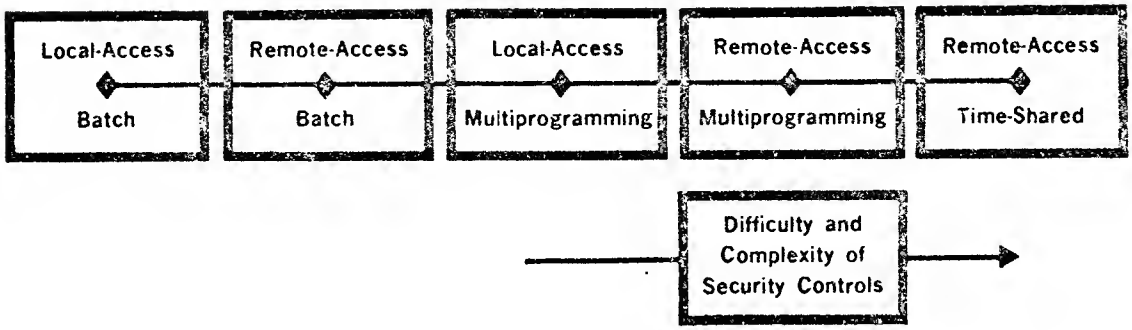

Figure 2-1

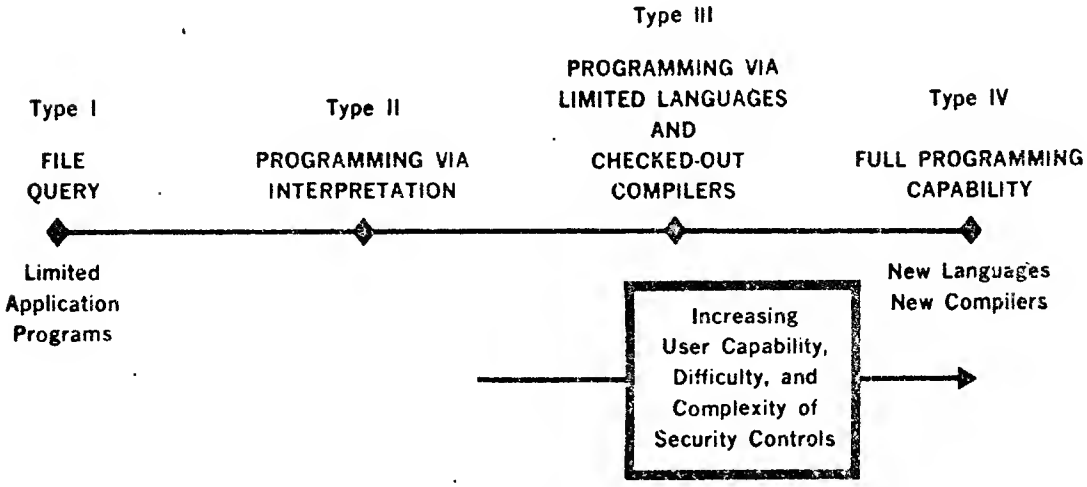

Figure 2-2

2.1.2.2 User Capabilities

"Another way of viewing the types of systems, shown in figure 2-2, is based on the levels of computing capability available to the user.

"File-query Systems (Type I) enable the user to execute only limited application programs embedded in the system and not available to him for alteration or change. He selects for execution one or more available application programs. He may be able to couple several of these programs together for automatic execution in sequence and to insert parameters into the selected programs.

"Interpretive systems (Type II) provide the user with a programming capability, but only in terms of input language symbols that result in direct execution within the computer of the operations they denote. Such symbols are not used to construct an internal machine language program that can subsequently be executed upon command from the user. Thus, the user cannot obtain control of the machine directly, because he is buffered from it by the interpretive software.

"Compiler systems (Type III) provide the user with a programming capability, but only in terms of languages that execute through a compiler embedded in the system. The instructions to the compiler are translated by it into an assembly language or basic machine language program. Program execution is controlled by the user; however, he has available to him only the limited compiler language.

"Full programming systems (Type IV) give the user extensive and unrestrained programming capability. Not only can he execute programs written in standard compiler languages, but he also can create new programming languages, write compilers for them, and embed them within the system. This gives the user intimate interaction with and control over the machine's complete resources--excepting of course, any resources prohibited to him by information-protecting safeguards (e.g., memory protection, base register controls, and I/O hardware controls).

In principle, all combinations of equipment configurations (figure 2-1) and operational capabilities (figure 2-2) can exist. In practice, not all the possible combinations have been implemented, and not all the possibilities would provide useful operational characteristics.

2.1.3 Threats To System Security

"By their nature, computer systems bring together a series of vulnerabilities. There are human vulnerabilities throughout; individual acts can accidentally or deliberately

jeopardize the system's information protection capabilities. Hardware vulnerabilities are shared among the computer, the communication facilities, and the remote units and consoles. There are software vulnerabilities at all levels of the machine operating system and supporting software; and there are vulnerabilities in the organization of the protection system (e.g., in access control, in user identification and authentication, etc.). How serious any one of these might be depends on the sensitivity (classification) of the information being handled, the class of users, the computational capabilities available to the user, the operating environment, the skill with which the system has been designed, and the capabilities of potential attackers of the system.

"These points of vulnerability are applicable both in industrial environments handling proprietary information and in government installations processing classified data. This Report is concerned directly with only the latter; it is sufficient here to acknowledge that the entire range of issues considered also has a "civil" side to which this work is relevant.

"The design of a secure system must provide protection against the various types of vulnerabilities. These fall into three major categories: accidental disclosures, deliberate penetrations, and physical attack.

"Accidental Disclosure. A failure of components, equipment, software, or subsystems, resulting in an exposure of information or violation of any element of the system. Accidental disclosures are frequently the result of failures of hardware or software. Such failures can involve the coupling of information from one user (or computer program) with that of an other user, the "clobbering" of information (i.e., rendering files or programs unusable), the defeat or circumvention of security measures, or unintended change in security status of users, files, or terminals. Accidental disclosures may also occur by improper actions of machine operating or maintenance personnel without deliberate intent.

"Deliberate Penetration. A deliberate and covert attempt to (1) obtain information contained in the system, (2) cause the system to operate to the advantage of the threatening party, or (3) manipulate the system so as to render it unreliable or unusable to the legitimate operator. Deliberate efforts to penetrate secure systems can either be active or passive. Passive methods include wire tapping and monitoring of electromagnetic emanations. Active infiltration is an attempt to enter the system so as to obtain data from the files or to interfere with data files or the system.

"One method of accomplishing active infiltration is for a legitimate user to penetrate portions of the system for which he has the authorization. The design problem is one of preventing access to files by someone who is aware of the access control mechanisms and who has the knowledge and desire to manipulate them to his own advantage. For example, if the access control codes are all four-digit numbers, a user can pick any four-digit number, and then, having gained access to some file, begin interacting with it in order to learn its contents.

"Another class of active infiltration techniques involves the exploitation of trap-door entry points in the system that by-pass the control facilities and permit direct access to files. Trap-door entry points often are created deliberately during the design and development stage to simplify the insertion of authorized program changes by legitimate system programmers, with the intent of closing the trap-door prior to operational use. Unauthorized entry points can be created by a system programmer who wishes to provide a means for bypassing internal security controls and thus subverting the system. There is also the risk of implicit trap-doors that may exist because of incomplete system design--i.e., loopholes in the protection mechanisms. For example, it might be possible to find an unusual combination of system control variables that will create an entry path around some or all of the safeguards.

"Another potential mode of active infiltration is the use of a special terminal illegally tied into the communication system. Such a terminal can be used to intercept information flowing between a legitimate terminal and the central processor, or to manipulate the system. For example, a legitimate user's sign-off signal can be intercepted and cancelled; then, the illegal terminal can take over interaction with the processor. Or, an illegal terminal can maintain activity during periods when the legitimate user is inactive but still maintaining an open line. Finally, the illegal terminal might drain off output directed to a legitimate terminal and pass on an error message in its place so as to delay detection.

"Active infiltration also can be by an agent operating within the secure organization. This technique may be restricted to taking advantage of system protection inadequacies in order to commit acts that appear accidental but which are disruptive to the system or to its users, or which could result in acquisition of classified information. At the other extreme, the agent may actively seek to obtain removable files or to create trap doors that can be exploited at a later date. Finally, an agent might be placed in the organization simply to learn about the system and the operation of the installation, and to obtain what

pieces of information come his way without any particularly covert attempts on his part at subversion.

"In passive subversion, means are applied to monitor information resident within the system or being transmitted through the communication lines without any corollary attempt to interfere with or manipulate the system. The most obvious method of passive infiltration is the wire tap. If communications between remote terminals and the central processor are over unprotected circuits, the problem of applying a wire tap to the computer line is similar to that of bugging a telephone call. It is also possible to monitor the electromagnetic emanations that are radiated by the high-speed electronic circuits that characterize so much of the equipment used in computational systems. Energy given off in this form can be remotely recorded without having to gain physical access to the system or to any of its components or communication lines. The possibility of successful exploitation of this technique must always be considered.

"Physical Attack. Overt assault against or attack upon the physical environment (e.g., mob action) is a type of vulnerability outside the scope of this Report.

2.1.4 Areas of Security Protection

"The system design must be aware of the points of vulnerability, which may be thought of as leakage points, and he must provide adequate mechanisms to counteract both accidental and deliberate events. The specific leakage points touched upon in the foregoing discussion can be classified in five groups: physical surroundings, hardware, software, communication links, and organizational (personnel and procedures). The overall safeguarding of information in a computer system, regardless of configuration, is achieved by a combination of protection features aimed at the different areas of leakage points. Procedures, regulations, and doctrine for some of these areas are already established within DoD, and are not therefore within the purview of the Task Force. However, there is some overlap between the various areas, and when the application of security controls to computer systems raises a new aspect of an old problem, the issue is discussed. An overview of the threat points is depicted in figure 2-3.

2.1.4.1 Physical Protection

"Security controls applied to safeguard the physical equipment apply not only to the computer equipment itself and to its terminals, but also to such removable items as printouts, magnetic tapes, magnetic disc packs, punch cards, etc. Adequate DoD regulations exist for dissemination,

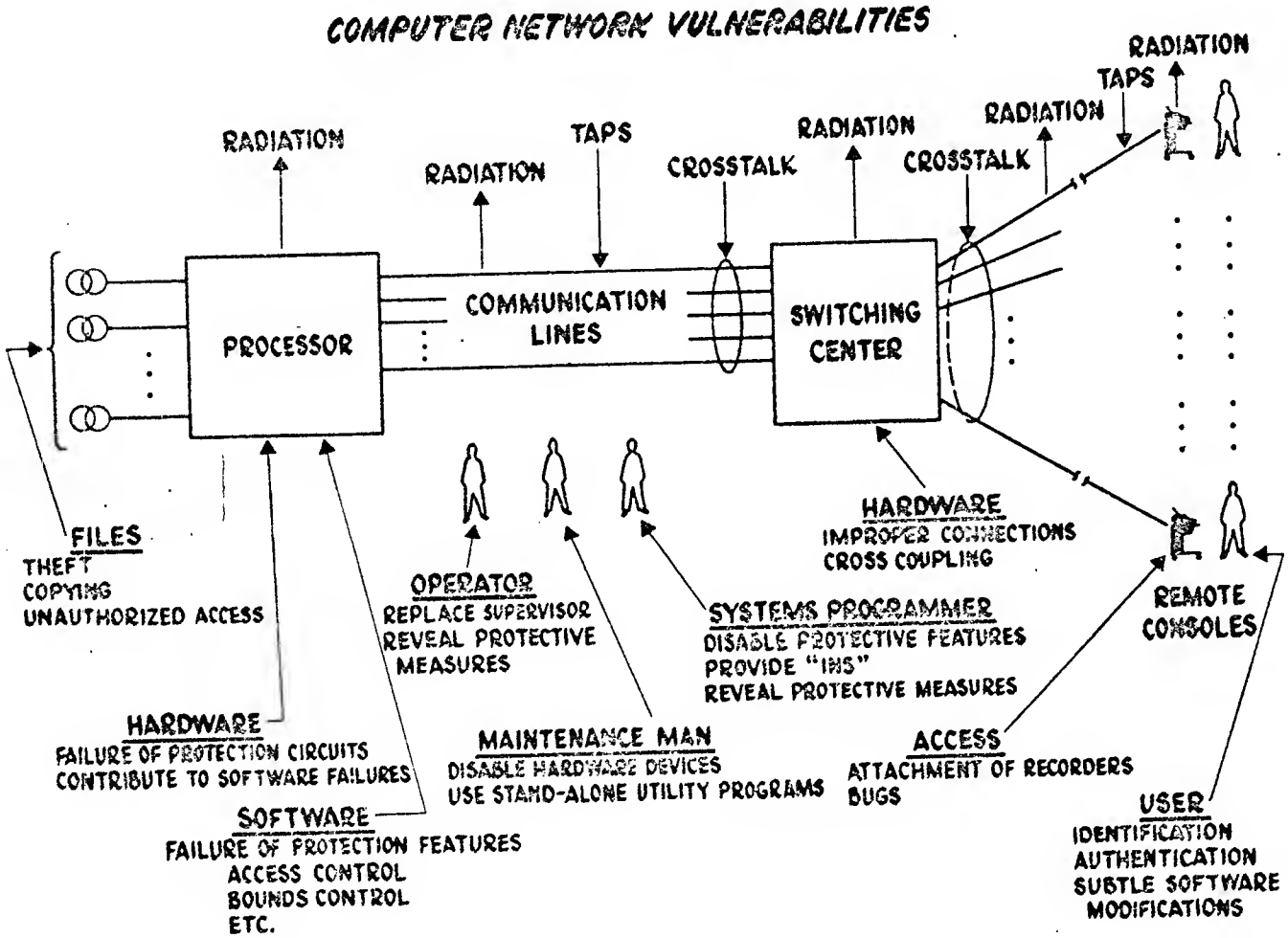

Figure 2–3

control, storage, and accountability of classified removable items. Therefore, security measures for these elements of the system are not examined in this Report unless there are some unique considerations. The following general guidelines apply to physical protection.

(a) The area containing the central computing complex and associated equipment (the machine room or operational area) must be secured to the level commensurate with the most highly classified and sensitive material handled by the system.

(b) Physical protection must be continuous in time, because, of the threat posed by the possibility of physical tampering with equipment and because of the likelihood that classified information will be stored within the computer system even when it is not operating.

(c) Remote terminal device must be afforded physical protection commensurate with the classification and sensitivity of information that can be handled through them. While responsibility for instituting and maintaining physical protection measures is normally assigned to the organization that controls the terminal, it is advisable for a central authority to establish uniform physical security standards (specific protection measures and regulations) for all terminals in a given system to insure that a specified security level can be achieved for an entire system. Terminal protection is important in order to:

- Prevent tampering with a terminal (installing intelligence sensors);

- Prevent visual inspection of classified work in progress;

- Prevent unauthorized persons from trying to call and execute classified programs or obtain classified data.

"If parts of the computer system (e.g., magnetic disc files, copies of printouts) contain unusually sensitive data, or must be physically isolated during maintenance procedures, it may be necessary to physically separate them and independently control access to them. In such cases, it may be practical to provide direct or remote visual surveillance of the ultra-sensitive areas. If visual surveillance is used, it must be designed and installed in such a manner that it cannot be used as a trap-door to the highly sensitive material it is intended to protect.

2.1.4.2 Hardware Leakage Points

"Hardware portions of the system are subject to malfunctions that can result directly in a leak or cause a failure of security protection mechanisms elsewhere in the system, including inducing a software malfunction. In addition, properly operating equipment is susceptible to being tapped or otherwise exploited. The types of failures that must directly affect security include malfunctioning of the circuits for such protections as bounds registers, memory read-write protect, privileged mode operation, or priority interrupt. Any hardware failure potentially can affect security controls; e.g., a single-bit error in memory.

"Both active and passive penetration techniques can be used against hardware leakage points. In the passive mode, the intervener may attempt to monitor the system by tapping into communications lines, or by monitoring compromising emanations. Wholly isolated systems can be physically shielded to eliminate emanations beyond the limits of the secure installation, but with geographically dispersed systems comprehensive shielding is more difficult and expensive. Currently, the only practical solutions are those used to protect communications systems.

"The problem of emanation security is covered by existing regulations; there are not new aspects to this problem raised by modern computing systems. It should be emphasized, however, that control of spurious emanations must be applied not only to the main computing center, but to the remote equipment as well.

"Although difficult to accomplish, the possibility exists that covert monitoring devices can be installed within the central processor. The problem is that the computer hardware involved is of such complexity that it is easy for a knowledgeable person to incorporate the necessary equipment in such a way as to make detection very difficult. His capability to do so assumes access to the equipment during manufacture or major maintenance. Equipment is also vulnerable to deliberate or accidental rewiring by maintenance personnel so that installed hardware appears to function normally, but in fact by-passes or changes the protection mechanisms.

"Remote consoles also present potential radiation vulnerabilities. Moreover, there is a possibility that recording devices might be attached to a console to pirate information. Other remote or peripheral equipment can present dangers. Printer ribbons or platens may bear impressions that can be analyzed; removable storage media (magnetic tapes, disc packs, even punch cards) can be stolen, or at least removed long enough to be copied.

"Erasure standards for magnetic media are not within the scope of this Task Force to review or establish. However, system designers should be aware that the phenomena of retentivity in magnetic materials is inadequately understood, and is a threat to system security.

2.1.4.3 Software Leakage Points

"Software leakage points include all vulnerabilities directly related to the software in the computer system. Of special concern is the operating system and the supplementary programs that support the operating system because they contain the software safeguards. Weaknesses can result from improper design, or from failure to check adequately for combinations of circumstances that can lead to unpredictable consequences. More serious, however, is the fact that operating systems are very large, complex structures, and thus it is impossible to exhaustively test for every conceivable set of conditions that might arise. Unanticipated behavior can be triggered by a particular user program or by a rare combination of user actions. Malfunctions might only disrupt a particular user's files or programs; as such, there might be no risk to security, but there is a serious implication for system reliability and utility. On the other hand, operating system malfunctions might couple information from one program (or user) to another; clobber information in the system (including information within the operating system software itself); or change classification of users, files, or programs. Thus, malfunctions in the system software represent potentially serious security risks. Conceivably, a clever attacker might establish a capability to induce software malfunctions deliberately; hiding beneath the apparently genuine trouble, an on-site agent may be able to tap files or to interfere with system operation over long periods without detection.

"The security safeguards provided by the operating system software include access controls, user identification, memory bounds control, etc. As a result of a hardware malfunction, especially a transient one, such controls can become inoperative. Thus, internal checks are necessary to insure that the protection is operative. Even when this is done, the simultaneous failure of both the protection feature and its check mechanism must always be regarded as a possibility. With proper design and awareness of the risk, it appears possible to reduce the probability of undetected failure of software safeguards to an acceptable level.

"Probably the most serious risk in system software is incomplete design, in the sense that inadvertent loopholes exist in the protective barriers and have not been foreseen by the designers. Thus, unusual actions on the part of users, or unusual ways in which their programs behave, can

induce a loophole. There may result a security breach, a suspension or modification of software safeguards (perhaps undetected), or wholesale clobbering of internal programs, data, and files. It is conceivable that an attacker could mount a deliberate search for such loopholes with the expectation of exploiting them to acquire information either from the system or about the system--e.g., the details of its information safeguards.

2.1.4.4 Communication Leakage Points

"The communications linking the central processor, the switching center and the remote terminals present a potential vulnerability. Wiretapping may be employed to steal information from land lines, and radio intercept equipment can do the same to microwave links. Techniques for intercepting compromising emanations may be employed against the communications equipment even more readily than against the central processor or terminal equipment. For example, crosstalk between communications lines or within the switching central itself can present a vulnerability. Lastly, the switch gear itself is subject to error and can link the central processor to the wrong user terminal.

2.1.4.5 Organizational Leakage Points

"There are two prime organizational leakage points, personnel security clearances and institutional operating procedures. The first concerns the structure, administration, and mechanism of the national apparatus for granting personnel security clearances. It is accepted that adequate standards and techniques exist and are used by the cognizant authority to insure the reliability of those cleared. This does not, however, relieve the system designer of a severe obligation to incorporate techniques that minimize the damage that can be done by a subversive individual working from within the secure organization. A secure system must be based on the concept of isolating any given individual from all elements of the system to which he has no need for access. In the past, this was accomplished by denying physical access to anyone without a security clearance of the appropriate level. In resource-sharing systems of the future, a population of users ranging from uncleared to those with the highest clearance levels will interact with the system simultaneously. This places a heavy burden on the overall security control apparatus to insure that the control mechanisms incorporated into the computer systems are properly informed of the clearances and restrictions applicable to each user. The machine system must be designed to apply these user access restrictions reliably.

"In some installations, it may be feasible to reserve certain terminals for highly classified or highly sensitive or restricted work, while other terminals are used exclusively for less sensitive operation. Conversely, in some installations any terminal can be used to any degree of classification or sensitivity, depending on the clearance and needs of the user at the given moment. In either of these cases, the authentication and verification mechanisms built into the machine system can be relied upon only to the degree that the data on personnel and on operational characteristics provided it by the security apparatus are accurate.

"The second element of organizational leakage points concerns institutional operating procedures. The consequences of inadequate organizational procedures, or of their haphazard application and unsupervised use, can be just as severe as any other malfunction. Procedures include the insertion of clearance and status information into the security checking mechanisms of the machine system , the methods of authenticating users and of receipting for classified information, the scheduling of computing operations and maintenance periods, the provisions for storing and keeping track of removable storage media, the handling of printed machine output and reports, the monitoring and control of machine-generated records for the security apparatus, and all other functions whose purpose is to insure reliable but unobtrusive operation from a security control viewpoint. Procedural shortcomings represent an area of potential weakness that can be exploited or manipulated, and which can provide an agent with innumerable opportunities for system subversion. Thus, the installation operating procedures have the dual function of providing overall management efficiency and of providing the administrative bridge between the security control apparatus and the computing system and its users.

"The Task Force has no specific comments to make with respect to personnel security issues, other than to note that control of the movement of people must include control over access to remote terminals that handle classified information, even if only intermittently. The machine room staff must have the capability and responsibility to control the movement of personnel into and within the central computing area in order to insure that only authorized individuals operate equipment located there, have access to removable storage media, and have access to any machine parts not ordinarily open to casual inspection.

2.1.4.6 Leakage Point Ecology

"In dealing with threats to system security, the various leakage points cannot be considered only individually.

Almost any imaginable deliberate attempt to exploit weaknesses will necessarily involve a combination of factors. Deliberate acts mounted against the system to take advantage of or to create leakage points would usually require both a system design shortcoming, either unforeseen or undetected, and the placement of someone in a position to initiate action. Thus, espionage activity is based on exploiting a combination of deficiencies and circumstances. A software leak may be caused by a hardware malfunction. The capability to tap or tamper with hardware may be enhanced because of deficiencies in software checking routines. A minor, ostensibly acceptable, weakness in one area, in combination with similar shortcomings in seemingly unrelated activities, may add up to a serious potential for system subversion. The system designer must be aware of the totality of potential leakage points in any system in order to create or prescribe techniques and procedures to block entry and exploitation.

"The security problem of specific computer systems must be solved on a case-by-case basis employing the best judgment of a team consisting of system programmers, technical, hardware, and communications specialists, and security experts."

2.2 TECHNOLOGY DEVELOPMENT HISTORY

Much has been learned about methods of assuring the integrity of information processed on computers since the emergence of operating systems in the early 1960s. Those early efforts were primarily concerned with improvements in the effective use of the large computer centers that were then being established. Information protection was not a major concern since these centers were operated as large isolated data banks. There were many significant hardware and software advances in support of the new operating system demands. Many of these changes were beneficial to the interests of information protection but since protection was not an essential goal at that time, the measures were not applied consistently and significant protection flaws existed in all commercial operating systems [TANG80].

In the late 1960s, spurred by activities such as the Defense Science Board study quoted in the previous section, efforts were initiated to determine how vulnerable computer systems were to penetration. The "Tiger Team" system penetration efforts are well known. Their complete success in penetrating all commercial systems attempted, provided convincing evidence that the integrity of computer systems hardware and software could not be relied upon to protect information from disclosure to other users of the same computer system.

By the early 1970s penetration techniques were well understood. Tools were developed to aid in the systematic detection of critical system flaws. Some detected mechanical coding errors, relying on the sophistication of the user to discover a way to exploit the flaws [ABBO76], others organized the search into a set of generic conditions which when present often indicated an integrity flaw [CARL75]. Automated algorithms were developed to search for these generic conditions, freeing the "penetrator" from tedious code searches and allowing the detailed analysis of specific potential flaws. These techniques have continued to be developed to considerable sophistication. In addition to their value in searching for flaws in existing software, these algorithms are useful as indicators of conditions to avoid in writing new software if one wishes to avoid the flaws that penetrators most often exploit.

These penetration aids are, however, of limited value in producing high integrity software systems. While they could be used to reveal certain types of flaws, they could assure the analysts that no further exploitable flaws of other types did not remain.

In the early 1970s the Air Force/Electronic Systems Division (ESD) conducted in-depth analyses of the requirements for

secure systems [ANDE72]. The concepts which emerged from their efforts are today the basis for most major trusted computer system developments. The basic concept is a Reference Monitor which mediates the access of all active system elements (people or programs) referred to as subjects, to all systems elements containing information (files, record, etc.) referred to as objects. All of the security relevant decision making functions within a conventional operating system are collected into a small primitive but complete operating system referred to as the Security Kernel. The security kernel is a specific implementation of the reference monitor in software and hardware. The three essential characteristics of this kernel are that it be:

> complete (i.e., that all accesses of all subjects to all objects be checked by the kernel);
>
> isolated (i.e., that the code that comprises the kernel be protected from modification or interference by any other software within the system);
>
> correct (i.e., that it perform the function for which it was intended and no other function).

Since these Air Force studies, considerable effort has gone into building security kernels for various systems. The reference monitor concept was the basis for work by MIT, MITRE and Honeywell in restructuring the Multics operating system [SCHR77]. MITRE and UCLA have built prototype security kernels for the PDP-11 minicomputer [WOOD77,POPE79]. System Development Corporation (SDC) is building a security kernel for the IBM VM/370 operating system [GOLD79]. Ford Aerospace and Communications Corporation is implementing the Kernelized Secure Operating System [MCCA79,BERS79] based on the Secure UNIX prototypes of UCLA and MITRE. AUTODIN II, the DCA secure packet switching system is employing this technology in the packet switching nodes. The Air Force SACDIN program (formerly called SATIN IV) is also employing this technology.

2.3 TRUSTED OPERATING SYSTEM FUNDAMENTALS

An operating system is a specialized set of software which provides commonly needed functions for user developed application programs. All operating systems provide a well defined interface to application programs in the form of system calls and parameters. Figure 2-4 illustrates the relationship between the operating system and application software. The operating system interfaces to the hardware through the basic machine instruction set and to applications software through the system calls which constitute the entry points to the operating system. Applications programs (e.g., A, B, and C) utilize these system calls to perform their specific tasks.

A trusted operating system patterned after an existing system is illustrated in figure 2-5. The security kernel is a primitive operating system providing all essential security relevant functions including process creating and execution and mediation of primary interrupt and trap responses. Because of the need to prove that the security relevant aspects of the kernel perform correctly, great care is taken to keep the kernel as small as possible. The kernel interface is a well defined set of calls and interrupt entry points. In order to map these kernel functions into a specific operating system environment, the operating system emulator provides the nonsecurity software interface for user application programs which is compatible with the operating system interface in figure 2-4. The level of compatibility determines what existing single security level application programs (e.g., A, B, C) can operate on the secure system without change.

Dedicated systems often do not need or cannot afford the facilities or environment provided by a general purpose operating system, but they may still be required to provide internal protection. Because the security kernel interface is well defined and provides all the primitive functions needed to implement an operating system it can be called directly by specialized application programs which provide their own environment in a form tailored for efficient execution of the application program. Examples of this type of use are dedicated data base management and message handling systems.

Figure 2-6 illustrates the relationship between two typical computer systems connected by a network. Each system is composed of an operating system (depicted by the various support modules arrayed around the outside of each box) and application programs (e.g., A, Q, and R in the inner area of the boxes). The dotted path shows how a terminal user on System I might access File X on System II. Working through the terminal handler, the user must first communicate with

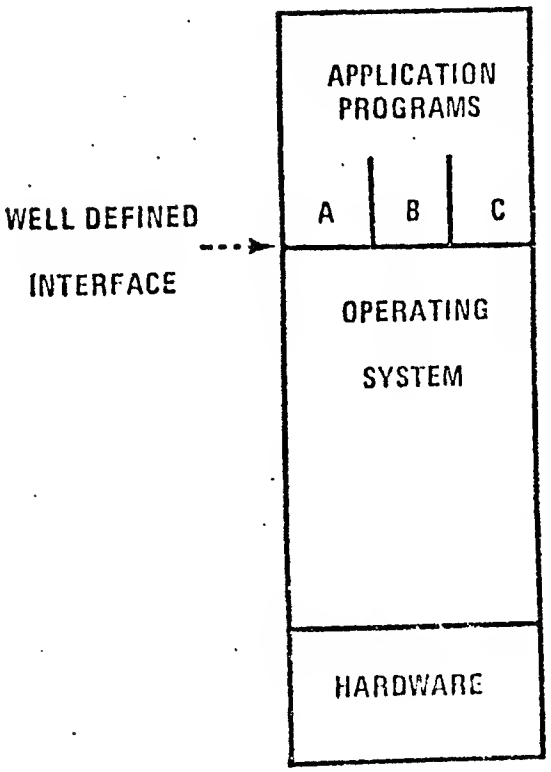

Figure 2-4

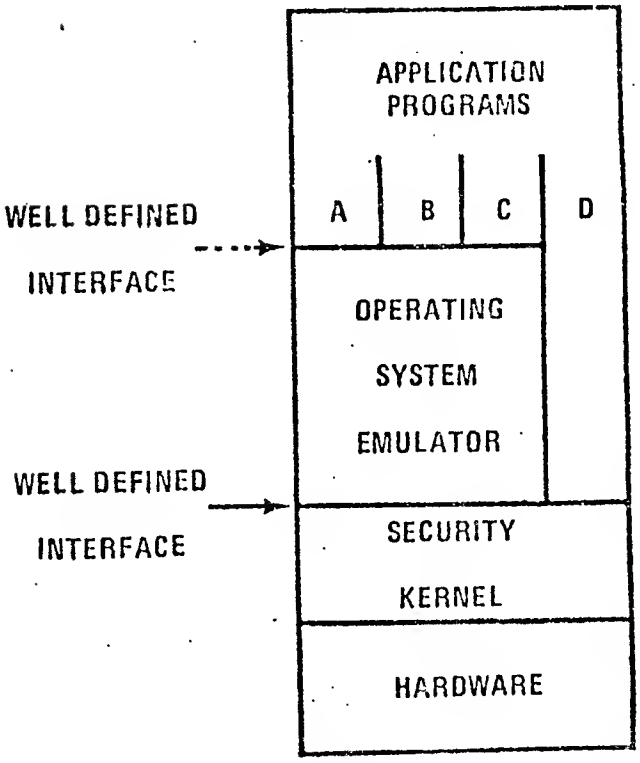

Figure 2-5

an application program (A) which will initiate a network connection with the remote computer through the network interface software. On System II an application program or a system utility (Q) is initiated on the user's behalf to access File X using the file system. Program Q could perform a data base update or retrieval for the user or it could arrange to transfer the file across the network to the local computer for processing.

When this scenario is applied in a secure environment, the two systems are placed in physically secure areas and, if the network is not secure, encryption devices are installed at the secure interface to the network as shown in figure 2-6.

Figure 2-7 illustrates the function of the security kernel in the above scenario. Because the kernel resides directly on the hardware (figure 2-5) and processes all interrupts, traps and other system actions, it is logically imposed between all "subjects" and "objects" on the system and can perform access checks on every event affecting the system. It should be noted that depending on the nature of the hardware architecture of the system, the representation of the kernel may have to include the various I/O device handlers. The DEC PDP-11, for example, requires that all device handlers be trusted and included in the kernel since I/O has direct access to memory. The Honeywell Level 6 with the Security Protection Module option (under development) does not require trusted device drivers since I/O access to memory is treated the same way as all other memory accesses and can be controlled by the existing hardware mechanisms.

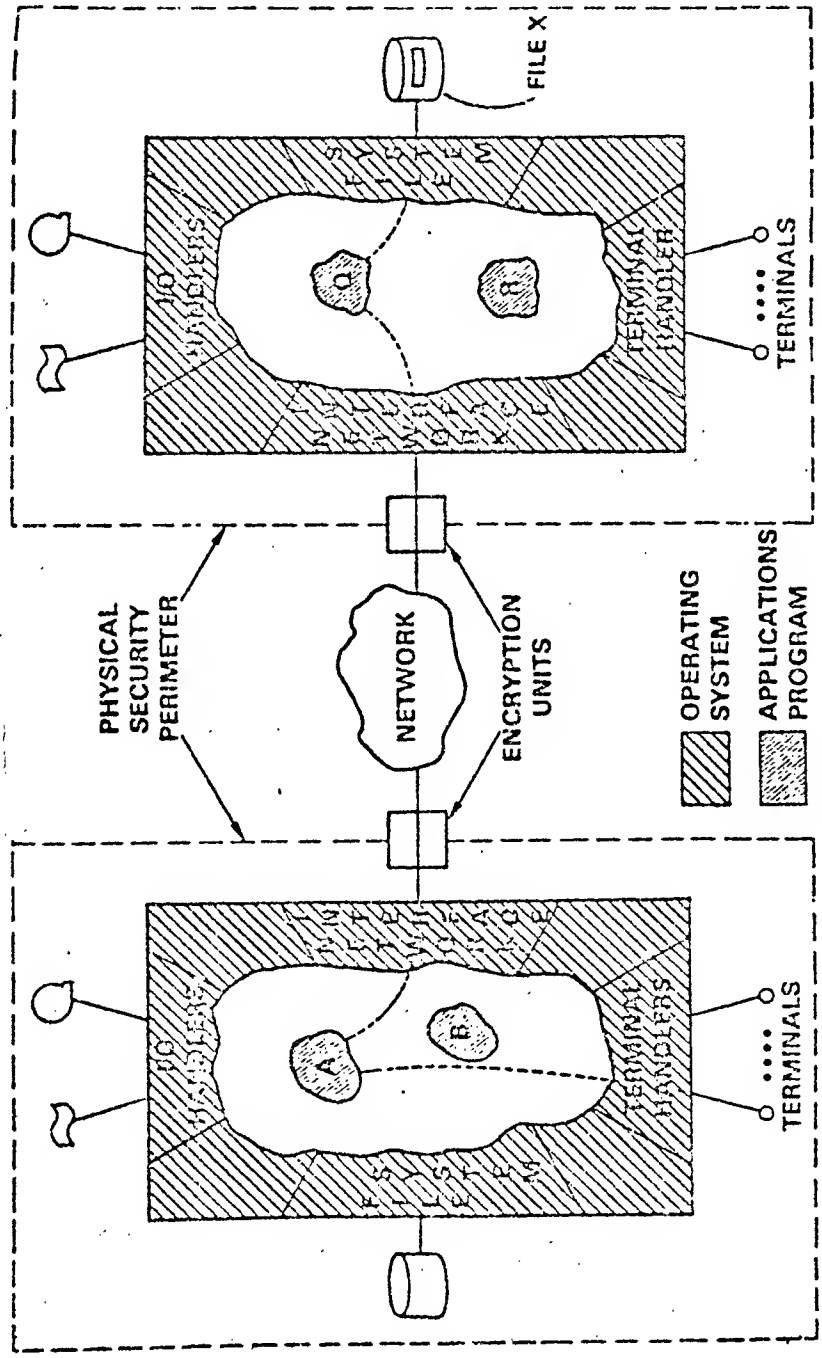

Figure 2-6

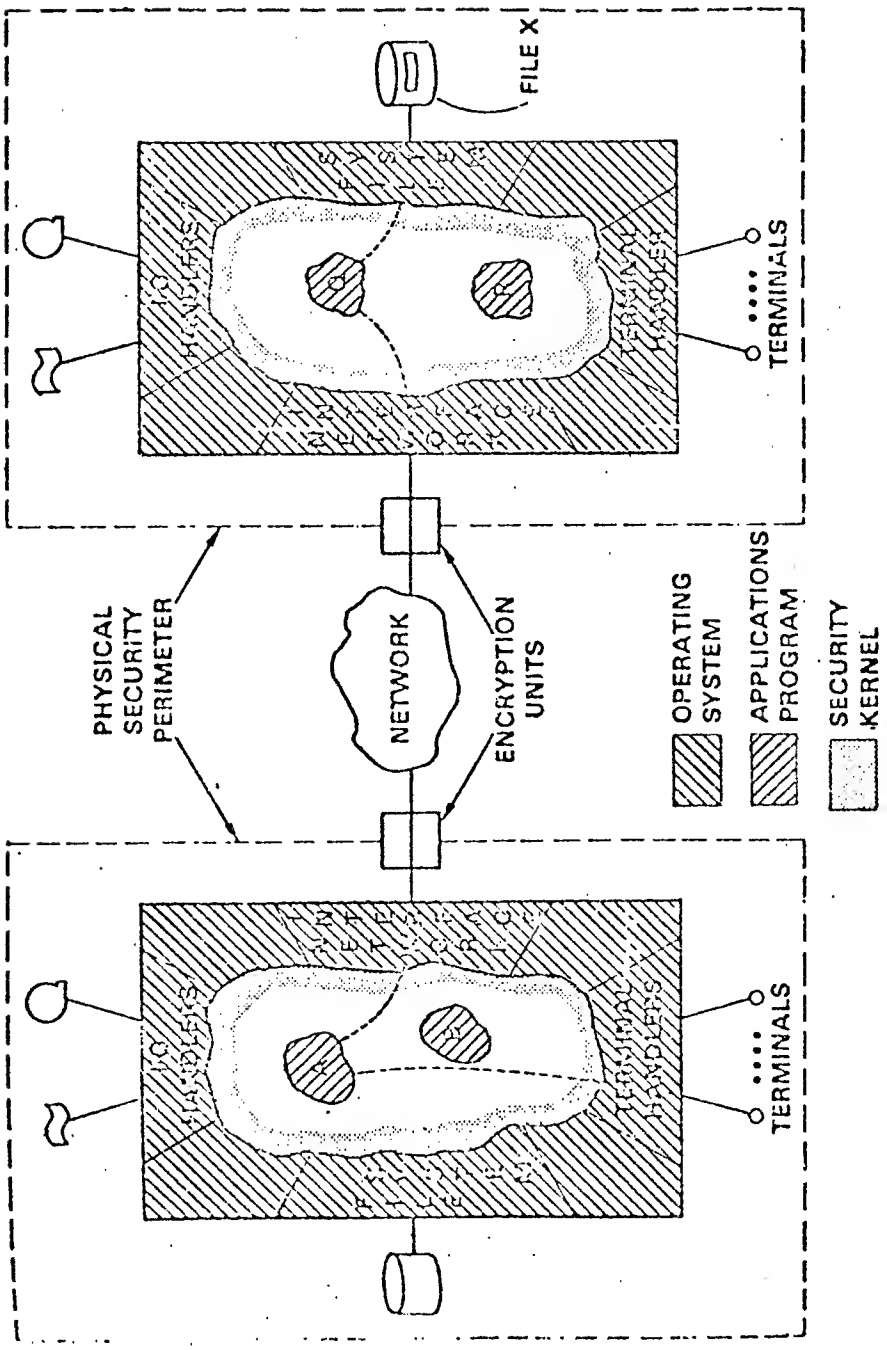

Figure 2-7

2.4 SYSTEM SECURITY VULNERABILITIES

Protection is always provided in relative quantities. Complete security is not possible with today's physical security measures, nor will it be with new computer security measures. There will always be something in any security system which can fail. The standard approach to achieving reliable security is to apply multiple measures in depth. Traditional locks and fences provide degrees of protection by delaying an intruder until some other protection mechanism such as a roving watchman can discover the attempted intrusion. With computer systems this "delay until detected" approach won't always work. Once an intruder knows about a security flaw in a computer system, he can generally exploit it quickly and repeatedly with minimal risk of detection.

Research on the security kernel approach to building trusted operating systems has produced a positive change in this situation. While absolute security cannot be achieved, the design process for trusted computer systems is such that one can examine the spectrum of remaining vulnerabilities and make reasonable judgments about the threats he expects to encounter and the impact that countermeasures will have on system performance.

A caution must be stated that the techniques described here do not diminish the need for physical and administrative security measures to protect a system from unauthorized external attack. The computer security/integrity measures described here allow authorized users with varying data access requirements to simultaneously utilize a computer facility. They provide this capability which relies upon the existing physical and administrative security measures rather than replacing them.

The nature of traditional physical and administrative security vulnerabilities encountered in the operation of computers with sensitive information is well understood. Only users cleared to the security level of the computer complex are allowed access to the system. With the advent of trusted computer systems allowing simultaneous use of computers by personnel with different security clearances and access requirements, an additional set of security vulnerabilities comes into play. Table 2-A describes one view of this new vulnerability spectrum as a series of categories of concern. Each of these concerns was not serious in previous systems because there was no need or opportunity to rely on the integrity of the computer hardware and software.

The first category is the Security Policy which the system must enforce in order to assure that users access only

Table 2-A Operating System Security Vulnerabilities

Category	Function	Vulnerability Resolution	Relative Security Risk
Security Policy	Establish security relationship between all system users, resources (e.g., DoD Security Policy)	Review	Moderate
System Specification	Establish policy relationship for each system module (e.g., Parnas I/O assertions)	For each module establish security assertions which govern activity	High
High Order Language Implementation	Transform System Specification Provisions for each module into (e.g. Fortran, PASCAL, C)	Manual or interactive validation that HOL obeys system spec	High
Machine Language Implementation	Transform HOL implementation into binary codes which are executed by hardware	Compiler Testing	Moderate
↑Software (Installation Independent)			
↓Hardware (Installation Dependent)			
Hardware Instruction Modules	Perform machine instructions (e.g., ADD instruction)	Testing, redundant checks of security relevant hardware.	Low -- except for security related hardware
Circuit Electronics	Perform basic logic functions which comprise instructions (e.g., AND, OR functions)	Maintenance Testing	Low
Device Physics	Perform basic electromagnetic functions which comprise basic logic function. (e.g. electron interaction)	Maintenance Testing	Very Low

Approved For Release 2007/06/01 : CIA-RDP83M00914R001800060009-2

authorized data. This policy consists of the rules which the computer will enforce governing the interactions between system users. There are many different policies possible ranging from allowing no one access to anyone else's information to full access to all data on the system. The DoD security policy (table 2-B) consists of a lattice relationship in which there are classification levels, typically Unclassified through Top Secret, and compartments (or categories) which are often mutually exclusive groupings [BELL74]. With this policy a partial ordering relationship is established in which users with higher personnel security clearance levels can have access to information at lower classification levels provided the users also have a "need to know" the information. The vulnerability concern associated with the security policy is assuming that the policy properly meets the total system security requirements.

The second general concern is the System Specification Level. Here the function of each module within the system and its interface to other modules is described in detail. Depending upon the exact approach employed, the system specification level may involve multiple abstract descriptions. The vulnerability here is to be able to assure that each level of the specification enforces the policy previously established.

The next vulnerability concern is the high level language implementation. This category constitutes the actual module implementation represented in a high order language (HOL) such as EUCLID or PASCAL. This vulnerability involves the assurance that the code actually obeys the specifications. The next concern on the vulnerability list is the machine code implementation which includes the actual instructions to be run on the hardware. The step from HOL implementation to machine code is usually performed by a compiler and the concern is to assure that the compiler accurately transforms the HOL implementation into machine language.

The next level of concern is that the hardware modules implementing the basic instructions on the machine perform accurately the functions they represent. Does the ADD instruction perform an ADD operation correctly and nothing else? Finally, the last concerns include the circuit electronics and more fundamental device physics itself. Do these elements accurately perform in the expected manner?

As can be seen by analyzing this vulnerability spectrum, some of the areas of concern are more serious than others. In particular, relatively little concern is given to circuit electronics and device physics since there is considerable confidence that these elements will perform as expected. There is a concern with hardware modules, though in general

TABLE 2-B -- DoD Security Policy

I. Non discretionary (i.e., levels established by national policy must be enforced).

Compartments

	A	B	C
Top Secret			
Secret			
Confidential			
Unclassified			

Partially Ordered Relationship

Top Secret > Secret > Confidential > Unclassified

Compartments A, B, C are mutually exclusive

Example:

User in Compartment B, level Secret can have access to all information at Secret and below (e.g., Confidential and Unclassified) in that compartment, but no access to information in Compartments A or C.

II. Discretionary, "Need to know" - (i.e., levels established "informally").

most nonsecurity relevant hardware failures do not pose a significant vulnerability to the security of the system and will be detected during normal operations of the machine. Those security relevant hardware functions can be subject to frequent software testing to insure (to a high degree) that they are functioning properly. The mapping between HOL and machine code implementation is a serious concern. The compiler could perform improper transformations which would violate the integrity of the system. This mapping can be checked in the future by verification of the compiler (presently beyond the state-of-the-art). Today we must rely on rigorous testing of the compiler.

The selection of the security policy which the system must support requires detailed analysis of the application requirements but is not a particularly complex process and can be readily comprehended so the level of concern is not too high for this category.

The system specification and HOL implementations are the two areas which are of greatest concern both because of the complex nature of these processes and the direct negative impact that an error in either has on the integrity of the system. Considerable research has been done to perfect both the design specification process and methods for assuring its correct HOL implementation [POPE78b,MILL76,FEIE77, WALK79,MILL79]. Much of this research has involved the development of languages and methodologies for achieving a complete and correct implementation [ROUB77,AMBL76,HOLT78].

As stated earlier this vulnerability spectrum constitutes a set of conditions in which the failure of any element may compromise the integrity of the entire system. In the high integrity systems being implemented today, the highest risk vulnerability areas are receiving the most attention. Consistent with the philosophy of having security measures in depth, it will be necessary to maintain strict physical and administrative security measures to protect against those lower risk vulnerabilities that cannot or have not yet been eliminated by trusted hardware/software measures. This will result in the continued need to have cleared operation and maintenance personnel and to periodically execute security checking programs to detect hardware failures. Over the next few years as we understand better how to handle the high risk vulnerabilities we will be able to concentrate more on the lower risk areas and consequently broaden the classes of applications in which these systems will be suitable.

Computer system security vulnerabilities constitute paths for passing information to unauthorized users. These paths can be divided into two classes: direct (or overt) and indirect (or covert) channels [LAMP73,LIPN75]. Direct paths

grant access to information through the direct request of a user. If an unauthorized user asks to read a file and is granted access to it, he has made use of a direct path. The folklore of computer security is filled with case histories of commercial operating systems being "tricked" into giving direct access to unauthorized data. Indirect or covert channels are those paths used to pass information between two user programs with different access rights by modulating some system resource such as a storage allocation. For example, a user program at one access level can manipulate his use of disk storage so that another user program at another level can be passed information through the number of unused disk pages.

Unauthorized direct access information paths can be completely eliminated by the security kernel approach since all objects are labeled with access information and the kernel checks them against the subject's access rights before each access is granted. The user who is interested only in eliminating unauthorized direct data access can achieve "complete" security using these techniques. Many environments in which all users are cleared and only a "need-to-know" requirement exists, can be satisfied by such a system.

Indirect data paths are more difficult to control. Some indirect channels can be easily eliminated, others can never be prevented. (The act of turning off the power to a system can always be used to pass information to users.) Some indirect channels have very high bandwidth (memory to memory speeds), many operate at relatively low bandwidth. Depending upon the sensitivity of the application, certain indirect channel bandwidths can be tolerated. In most cases external measures can be taken to eliminate the utility of an indirect channel to a potential penetrator.

The elimination of indirect data channels often affects the performance of a system. This situation requires that the customer carefully examine the nature of the threat he expects and that he eliminate those indirect paths which pose a real problem in his application. In a recent analysis, one user determined that indirect path bandwidths of approximately teletype speed are acceptable while paths that operate at line printer speed are unacceptable. The assumption was that the low speed paths could be controlled by external physical measures. With these general requirements to guide the system designer it is possible to build a useful trusted system today.

2.5 TRUSTED SYSTEM ENVIRONMENTS

The applications for which trusted operating systems will be used and the environments in which they will operate cover a wide spectrum. The most sensitive practical environment encompasses highly sensitive intelligence information on a system with unclassified users. AUTODIN II is employing security kernel technology to operate a packet switched network in such an environment. A minimum sensitive environment in which a trusted system might be placed involves unclassified information where individual need-to-know or privacy must be maintained. There are a large number of environments between these two that have differing degrees of sensitivity.

The type of application for which the trusted system will be used influences the concern for the integrity of the system. For example, while AUTODIN II does not employ full code verification or fault resistant hardware, it is being used for an application which offers the user few opportunities to exploit weaknesses within the packet switch software. Thus it can be used in a much higher-risk environment than can a general-purpose computer system. A general-purpose programming environment offers many more opportunities to exploit system weaknesses. The combination of the sensitivity of information being processed relative to the clearances of the users and the degree of user capability afforded by a particular application are the primary factors in determining the level of concern required for a particular system.

There are examples of multilevel systems that have been approved which provide significant data points in the environment/application spectrum. Honeywell Multics, enhanced by an "access isolation mechanism", is installed as a general-purpose timesharing system at the Air Force Data Services Center in the Pentagon in a Top Secret environment with some users cleared only to the Secret level. Multics has the best system integrity of any commercial operating system available today. While it does not have formal design specifications as described in the previous section, the system was designed and structured with protection as a major goal but. Formal development procedures were not used because the system was developed before these techniques were available. In spite of this, after a very thorough and careful review, the Air Force determined that the benefit of using this system exceeded the risk that a user might attempt to exploit a system weakness, given that all users have at least a Secret clearance.

There have been several other examples where current technology enhanced by audit procedures and subjected to rigorous testing have been approved for use in limited

sensitivity applications.

The degree to which one must rely on technical features of a system for integrity depends significantly on the environment that the system will operate in and the capabilities that a user has to exploit system weaknesses. There has been some study of the range of sensitivities for different applications and environments [ADAM79]. Section 3.1 describes a way of combining these application and environment concerns with the technical measures of system integrity.

2.6 VERIFICATION TECHNOLOGY OVERVIEW

The security kernel approach to designing trusted computing systems collects the security relevant portions of the operating system into a small primitive operating system. In order to have confidence that the system can be trusted, it is necessary to have confidence that the security kernel operates correctly. That is, one must have confidence that the security kernel enforces the security policy which the system is supposed to obey.

Traditional means such as testing and penetration can and should be used to uncover flaws in the security kernel implementation. Unfortunately, it is not possible to test all possible inputs to a security kernel. Thus, although testing may uncover some flaws, no amount of testing will guarantee the absence of flaws. For critical software, such as a security kernel, additional techniques are needed to gain the necessary assurance that the software meets its requirements. Considerable research has been devoted to techniques for formally proving that software operates as intended. These techniques are referred to as software verification technology or simply verification technology.

In the case of a security kernel, the critical aspect of its operation is the enforcement of a security policy. The ultimate goal of a verification is to prove that the implemented security kernel enforces the desired security policy. There are five main areas of concern in relating the security policy to the implemented security kernel: the security policy itself, system specification, high order language implementation, compiler, and hardware. The following paragraphs discuss the way in which verification addresses each of these areas.

2.6.1 Security Policy

DoD has established regulations covering the handling of classified information (e.g. DOD Directive 5200.28). However, in order to prove that a security kernel enforces a security policy, it is necessary to have a formal mathematical model of the security policy. It is not possible to prove that the model is correct since the model is a formal mathematical interpretation of a non-mathematical policy. Fortunately, mathematical models of security have existed since 1973 when Bell and LaPadula formulated a model of multilevel security. [BELL74]. Various models of multilevel security have been used since 1973, but they have all been derived from the original Bell-LaPadula model. Since this model has been widely disseminated and discussed, one can have confidence that the model correctly reflects the non-mathematical DoD regulations. In the case of software with security

requirements different from those of a security kernel, a specialized model is needed, and thorough review is required to determine that the model guarantees the informal requirements.

2.6.2 System Specification

In practice the gap between the mathematical model of security and the implemented security kernel is too great to directly prove that the kernel enforces the model. A specification of the system design can be used to break the proof up into two parts:

a) Show the system specification obeys the model.

b) Show the kernel code correctly implements the specification.

Step a) is called Design Verification. Step b) is called Implementation or Code Verification.

To be useful for verification, the meaning of the system specification must be precisely defined. This requires that a formally defined specification language be used. Formal specification languages, associated design and verification methodologies, and software tools to help the system designer and verifier have been developed by several organizations. Since a specification typically hides much of the detail which must be handled in an implementation, design verification is significantly easier than code verification. The design verification usually requires the proof of a large number of theorems, but most of these theorems can be handled by automatic theorem provers. There are several methodologies available today that work with existing automatic theorem provers. Verification that a formal design specification obeys a security model has been carried out as part of the AUTODIN II, SACDIN, KSOS, and KVM/370 programs. Design verification can be useful even if no code verification is done. Traditional techniques can give some confidence that the code corresponds to the implementation, and design verification will uncover design flaws, which are the most difficult to correct.

2.6.3 HOL, Compiler, Hardware

After the system specification has been verified to obey the security model, the remaining problem is to show that the kernel implementation is consistent with its specification. The gap from specification to object code is too great for current verification methodologies to prove that object code is consistent with a specification. However, work has been devoted to developing techniques for proving that the HOL implementation of a system is consistent with its

specification. The implementation for a system is much more detailed than a specification, and more attributes must be shown to be true to support the top-level design assertions. Thus, verification that the code is consistent with its specification is much more difficult than verification of the design properties of the specification. Usually many theorems must be proved for code verification. Even with automatic theorem provers the verification requires significant human and computer resources. Recent work in verification technology has developed code verification to the point that it is now feasible to attempt code verification in some small systems. To date, code verification has been done only for example systems.

To complete the verification one would have to consider the compiler and hardware. At present, it is beyond the state of the art to formally prove that production compilers or hardware operate as specified. However, since the compiler and hardware will probably be used on many systems, flaws in their operation are more likely to be revealed than flaws in the code for a new system. The software is the area where there is the greatest need for quality assurance effort.

2.6.4 Summary

Verification is useful for increasing one's confidence that critical software obeys its requirements. An example of critical software where verification can be useful is a security kernel. Verification does not show that a system is correct in every respect. Rather verification involves showing consistency between a mathematical model, a formal specification, and an implementation. Verification that a formal specification is consistent with a mathematical model of security has been demonstrated on several recent systems. Verification of consistency between a specification and a HOL implementation is on the verge of becoming practical for small systems, but has not yet been demonstrated except for example systems. Verification of consistency between the HOL and machine language is not practical in the near future. (Verification is discussed in more detail in section 4.3.)

SECTION 3

COMPUTER SECURITY INITIATIVE STATUS

The goal of the Computer Security Initiative is to establish widespread availability of trusted computer systems. There are three major activities of the Initiative seeking to advance this goal: (1) coordination of DoD R&D efforts in the computer security field, (2) identification of consistent and efficient evaluation procedures for determining suitable environments for the use of trusted computer systems, and (3) encouragement of the computer industry to develop trusted systems as part of their standard product lines. This section describes the Initiative activities in support of 2 and 3 above. (Section 4 addresses item 1.)

3.1 THE EVALUATED PRODUCTS LIST

Section 1-1101 of the Defense Acquisition Regulations (DAR, formerly called the Armed Services Procurement Regulations or ASPRS) defines a procedure for evaluating a product prior to a procurement action. This procedure establishes a Qualified Products List (QPL) of items which have met a predefined government specification. This procedure can be used when one or more of the following conditions exist:

> "(i) The time required to conduct one or more of the examinations and tests to determine compliance with all the technical requirements of the specification will exceed 30 days (720 hours). (Use of this justification should advance product acceptance by at least 30 days (720 hours).)
>
> (ii) Quality conformance inspection would require special equipment not commonly available.
>
> (iii) It covers life survival or emergency life saving equipment. (See 1-1902(b)(ii).)"

Whenever any of these conditions exist, a Qualified Products List process may be established. Under these regulations, a specification of the requirements that a product must meet is developed and widely distributed. Any manufacturer who believes his product meets this specification may submit his product for evaluation by the government. If the product is determined to meet the specification, it is entered on a Qualified Products List maintained by the government agency performing the evaluation.

Any agency or component seeking to procure an item which meets the QPL specification can utilize the QPL evaluation in its procurement process in lieu of performing its own separate evaluation. The QPL process allows the efficient and consistent evaluation of complex products and the general availability of the evaluation results to all DoD procurement organizations.

There is a provision of the QPL process described in the DAR that requires all products considered as part of a particular government RFP to be already on the QPL prior to issuance of the RFP. If a manufacturer believes that his product meets the government specification but the evaluation has not been completed at the time of issuance of the RFP, that product will be disqualified from that procurement action. This provision has been viewed by many as anti-competitive and has been a deterrent to the wide use of the QPL process.

The Special Committee on Compromising Emanations (SCOCE) of the National Communications Security Board has established a modified QPL process for the evaluation of industry devices which meet government standards for compromising emanations (NACSEM 5100). Under the provisions of their Preferred Products List (PPL), a manufacturer supplies the government with the results of tests performed either by himself or one of a set of industry TEMPEST evaluation laboratories which indicate compliance with the NACSEM 5100 specification. Upon affirmative review of these test results, the product will be entered on the TEMPEST Preferred Products List. Any manufacturer may present the results of the testing of his product to the government at any time including during the response to a particular RFP.

The evaluation of the integrity of industry developed computer systems is a complex process requiring considerable time and resources that are in short supply. A QPL-like process for disseminating the results of these evaluations is essential. Under these circumstances, a small team of highly competent government computer science and system security experts will perform the evaluation of industry submitted systems and the results of their evaluations will be made available to any DoD organization for use in their procurement process, eliminating the inefficiency and inconsistency of duplicate evaluations.

As described in section 3.4.1, there are many technical features which influence the overall integrity of a system. Some of these features are essential for protecting information within a system regardless of the type of application or the environment. However, many of these features may not be particularly relevant in particular applications or environments and therefore it may be

reasonable to approve systems for use in some environments even with known deficiencies in certain technical areas.

For example, in an environment where all users are cleared to a high level and there is a need-to-know requirement, it may be reasonable to employ a system which has not completely eliminated all indirect data paths (see section 2.4.1) on the premise that a high degree of trust has already been placed in the cleared users and they are not likely to conspire with another user to attempt to exploit a complex indirect channel to obtain information for which they have no need-to-know. Similar arguments can be made for systems processing information of a low level of sensitivity. Since indirect paths require two conspiring users they are difficult to use and in most cases are not worth the risk of being detected.

Thus, systems with certain technical features should be usable for applications of a particular type in environments of a particular type. It is possible to describe classes of those integrity features required for different application and risk environments. If there is a process (as described in section 3.4) for evaluating the integrity of various trusted systems, then an "Evaluated Products List" (EPL) can be constructed matching products to these protection classes (and, thus, to certain application and risk environments).

It appears that the technical integrity measures can be categorized into a small set of classes (six to nine) with considerable consistency in determining into which class a particular system will fit. Figure 3-1 is an example of an Evaluated Products List, consisting of six classes ranging from systems about which very little is known and which can be used only in dedicated system-high environments (most of the commercial systems today) to systems with technical features in excess of the current state-of-the-art. The environments are described in terms of the sensitivity of the information and the degree of user capability.

The Evaluated Products List includes all computer systems whose protection features have been evaluated. The first class implies superficial protection mechanisms. A system in this class is only suitable for a system-high classification installation. Most modern commercial systems satisfy at least the requirements of Class I. As one progresses to higher classes, the technical and assurance features with respect to system protection are significantly strengthened and the application environment into which a system may be placed can be of a higher sensitivity.

In discussing the Evaluated Products List (EPL) concept with various communities within the defense department and the intelligence community, it has become clear that, while the

Class	Technical features	Examples	Possible Environments
1	May have some protection Login authentication	Most modern commercial systems	Dedicated mode
2	Mandatory data security Penetration testing Auditing	Mature "enhanced" OS	Benign need-to-know environments
3	Top-Level Specification of TCB Clearly Identified & Protected TCB Top-Down Design Testing Based on TLS	Multics	AFDSC TS-S
4	Formal Top-Level Specifications TLS Verification Testing Generation from Formal TLS Limited Covert Path Provisions	KSOS-6 KSOS-11 KVM	Limited user programming TS-S-C
5	Verified Implementation Test Case Generation from LLS Extended Covert Path Provisions		Full user programming TS-S-C
6	Object Code Analysis Hardware Specs		Full user programming TS-S-C-U

Figure 3-1 EVALUATED PRODUCTS LIST

Approved For Release 2007/06/01 : CIA-RDP83M00914R001800060009-2

technical feature evaluation process is understood and agreed upon, the identification of suitable application environments will differ depending upon the community involved. For example, the Genser community may decide that the technical features of a Class IV system are suitable for a particular application, whereas the same application in the intelligence community may require a Class V system. As a result, the EPL becomes a matrix of suitable application environments (figure 3-2), depending upon the sensitivities of the information being processed. In addition to the intelligence community and the Genser community, there are the interests of the privacy and the financial communities and the non-national security communities whose requirements, frequently, are less restrictive than those of the national security communities.

The successful establishment of an Evaluated Products List for trusted computing systems requires that the computer industry become cognizant of the EPL concept and of computer security technology, and that a procedure for evaluating systems be formulated. Section 3.2 (below) discusses the focus of operating system protection requirements, the Trusted Computing Base. Section 3.3 describes the Initiative's technology transfer activities. Section 3.4 presents a proposed process for trusted system evaluation and section 3.5 summarizes current, informal system evaluation activity.

EVALUATED PRODUCTS LIST

CLASS	TECHNICAL FEATURES	INDUSTRY PRODUCTS	SUITABLE APPLICATION/ ENVIRONMENTS		
			INTELLIGENCE COMMUNITY	GENSER COMMUNITY	PRIVACY FINANCIAL NON-NAT'L SEC
1	—	—	—	—	—
2	—	—	—	—	—
3	—	—	—	—	—
4	—	—	—	—	—
5	—	—	—	—	—
6	—	—	—	—	—

FIGURE 3-2

Approved For Release 2007/06/01 : CIA-RDP83M00914R001800060009-2

3.2 THE TRUSTED COMPUTING BASE

A significant prerequisite to achieving the widespread availability of commercial trusted systems is the definition of just what the requirements for a trusted system are. Security kernel prototypes had been built over the years, but they were specific to particular hardware bases or operating systems. In order to present the basic concept of a security kernel and trusted processes in a general manner that would apply to a wide range of computer systems and many applications, a proposed specification for a Trusted Computing Base (a kernel and trusted processes) was prepared by Grace Nibaldi of The MITRE Corporation [NIBA79a]. The specification describes the concept of a Trusted Computing Base (TCB) and discusses TCB requirements. The rest of this section describes the Trusted Computing Base, and is excerpted from [NIBA79a]. (We have preceded the section numbering used in [NIBA79a] by TCB. Thus, Nibaldi's section 3.1 appears below as TCB.3.1.)

TCB.1 Scope

In any computer operating system that supports multiprogramming and resource sharing, certain mechanisms can usually be identified as attempting to provide protection among users against unauthorized access to computer data. However, experience has shown that no matter how well-intentioned the developers, traditional methods of software design and production have failed to provide systems with adequate, verifiably correct protection mechanisms. We define a trusted computing base (TCB) to be the totality of access control mechanisms for an operating system.

A TCB should provide both a basic protection environment and the additional user services required for a trustworthy turnkey system. The basic protection environment is equivalent to that provided by a security kernel (a verifiable hardware/software mechanism that mediates access to information in a computer system); the user services are analogous to the facilities provided by trusted processes in kernel-based systems. Trusted processes are designed to provide services that could be incorporated in the kernel but are kept separate to simplify verification of both kernel and trusted processes. Trusted processes also have been referred to as "privileged," "responsible," "semi-trusted", and "non-kernel security-related (NKSR)" in various implementations. This section documents the performance, design, and development requirements for a TCB for a general-purpose operating system.

In this section, there will be no attempt to specify how any particular aspect of a TCB must be implemented. Studies of

present-day computer architectures [SMIT75,TANG78] indicate that in the near term a significant amount of software will be needed for protection regardless of any support provided by the underlying hardware. In future computer architectures, more of the TCB functions may be implemented in hardware or firmware. Examples of specific hardware or software implementations are given merely as illustrations, and are not meant to be requirements.

This specification is limited to computer hardware and software protection mechanisms; not covered are the administrative, physical, personnel, communications, and other security measures that complement the internal computer security controls. For more information in those areas, see DoD Directive 5200.28 that describes the procedures for the Department of Defense.

(Section 2 of the TCB specification contains references. They have been included in the references for this report rather than being included here as TCB.2.)

TCB.3 General Requirements

TCB.3.1 System Definition

A TCB is a hardware and software access control mechanism that establishes a protection environment to control the sharing of information in computer systems. Under hardware and software we include implementations of computer architectures in firmware or microcode. A TCB is an implementation of a reference monitor, as defined in [ANDE72], that controls when and how data is accessed.

In general, a TCB must enforce a given protection policy describing the conditions under which information and system resources can be made available to the users of the system. Protection policies address such problems as undesirable disclosure and destructive modification of information in the system, and harm to the functioning of the system resulting in the denial of service to authorized users.

Proof that the TCB will indeed enforce the relevant protection policy can only be provided through a formal, methodological approach to TCB design and verification, an example of which is discussed below. Because the TCB consists of all the security-related mechanisms, proof of its validity implies the remainder of the system will perform correctly with respect to the policy.

Ideally, in an implementation, policy and mechanism can be kept separate so as to make the protection mechanisms flexible and amenable to different environments, e.g., military, banking, or medical applications. The advantage

here is that a change in or reinterpretation of the required policy need not result in rewriting or reverifying the TCB.

In the following sections, general requirements for TCB design and verification are discussed.

TCB.3.2 Protection Policy

The primary requirement on a TCB is that it support a well-defined protection policy. The precise policy will be largely application and organization dependent. Four specific protection policies are listed below as examples around which TCBs may be designed. All are fairly general purpose, and when used in combination, would satisfy the needs of most applications, although they do not specifically address the denial of service threat. The policies are ordered by their concern either with the viewing of information--security policies--or with information modification--integrity policies; and by whether the ability to access information is externally predetermined--mandatory policies--or controlled by the processor of the information--discretionary policies:

1. mandatory security (used by the Department of Defense--see DoDD 5200.28), to address the compromise of information involving national security;

2. discretionary security (commonly found in general purpose computer systems today);

3. mandatory integrity; and

4. discretionary integrity policy.

In each of these cases, "protection attributes" are associated with the protectable entities, or "objects" (computer resources such as files and peripheral devices that contain the data of interest), and with the users of these entities (e.g., users, processes), referred to as subjects. In particular, for mandatory security policy, the attributes of subjects and objects will be referred to as "security levels." These attributes are used by the TCB to determine what accesses are valid. The nature of these attributes will depend on the applicable protection policy.

See Nibaldi [NIBA79b] for a general discussion on policy. See Biba [BIBA75] for a discussion of integrity.

TCB.3.3 Reference Monitor Requirements

As stated above, a TCB is an implementation of a reference monitor. The predominant criteria for a sound reference

monitor implementation are that it be

1. complete in its mediation of access to data and other computer resources;

2. self-protecting, free from interference and spurious modification; and

3. verifiable, constructed in a way that enables convincing demonstration of its correctness and infallibility.

TCB.3.3.1 Completeness

The requirement that a TCB mediate every access to data in the computer system is crucial. In particular, a TCB should mediate access to itself--its code and private data--thereby supporting the second criterion for self-protection. The implication is that on every action by subjects on objects, the TCB is invoked, either explicitly or implicitly, to determine the validity of the action with respect to the protection policy. This includes:

1. unmistakably identifying the subjects and objects and their protection attributes, and

2. making it impossible for the access checking to be circumvented.

In essence, the TCB must establish an environment that will simultaneously (a) partition the physical resources of the system (e.g., cycles, memory, devices, files) into "virtual" resources for each subject, and (b) cause certain activities performed by the subjects, such as referencing objects outside of their virtual space, to require TCB intervention.

TCB.3.3.1.1 Subject/Object Identification

What are the subjects and objects for a given system and how are they brought into the system and assigned protection attributes? In the people/paper world, people are clearly the subjects. In a computer, the process has commonly been taken as a subject in security kernel-based systems, and storage entities (e.g., records, files, and I/O devices) are usually considered the objects. Note that a process might also behave as an object, for instance if another process sends it mail (writes it). Likewise, an I/O device might be considered to sometimes act as a subject, if it can access any area of memory in performing an operation. In any case, the policy rules governing subject/object interaction must always be obeyed. The precise breakdown for a given system will depend on the application. Complete identification of subjects and objects within the computer system can only be

assured if their creation, name association, and protection attribute assignment always take place under TCB control, and no subsequent manipulations on subjects and objects are allowed to change these attributes without TCB involvement. Certain issues remain, such as (a) how to associate individual users and the programs they run with subjects; and (b) how to associate all the entities that must be accessed on the system (i.e., the computer resources) with objects. TCB functions for this purpose are described in TCB.4, "Detailed Requirements."

TCB.3.3.1.2 Access Checking

How are the subjects constrained to invoke the TCB on every access to objects? Just as the TCB should be responsible for generating and unmistakably labelling every subject and object in the system, the TCB must also be the facility for enabling subjects to manipulate objects, for instance by forcing every fetch, store, or I/O instruction executed by non-TCB software to be "interpreted" by the TCB.

Hardware support for checking on memory accesses exists on several machines, and has been found to be very efficient. This support has taken the form of descriptor-based addressing: each process has a virtual space consisting of segments of physical memory that appear to the process to be connected. In fact, the segments may be scattered all over memory, and the virtual space may have holes in it where no segments are assigned. Whenever the process references a location, the hardware converts the "virtual address" into the name of a base register (holding the physical address of the start of the segment, the length of the segments, and the modes of access allowed on the segment), and an offset. The content of the base register is called a descriptor. The hardware can then abort if the form of reference (e.g., read, write) does not correspond to the valid access modes, if the offset exceeds the size of the segment, or if no segment has been "mapped" to that address. The software portion of the TCB need merely be responsible for setting up the descriptor registers based on one-time checks as to the legality of the mapping.

Access checking in I/O has been aided by hardware features in a variety of ways. In one line of computers, devices are manipulated through the virtual memory mechanism: a process accesses a device by referencing a virtual address that is subsequently changed by hardware into the physical address of the device. This form of I/O is referred to as "mapped I/O" [TANG78]. Other methods of checking I/O are discussed in section TCB.4.1.2.

TCB.3.3.2 Self-Protection

Following the principle of economy of mechanism [SALT75], the TCB ideally protects itself in the same way that it protects other objects, so the discussion on the completeness property applies here as well. In addition, not uncommonly many computer architectures provide for multiple protection "domains" of varying privilege (e.g., supervisor, user). Activities across domains are limited by the hardware so that software in the the more privileged domains might affect the operations in less privileged domains, but not necessarily vice versa. Also, software not executing in a privileged domain is restricted, again by the hardware, from using certain instructions, e.g., manipulate-descriptor-registers, set-privilege-bit, halt, and start-I/O. Generally only TCB software would run in the most privileged domain and rely on the hardware for its protection. (Of course, part of the TCB might run outside of that domain, e.g., as a trusted process.) Clearly, if in addition to the TCB, non-TCB or untrusted software were allowed to run in the privileged region, TCB controls could be subverted and the domain mechanism would be useless.

TCB.3.3.3 Verifiability

The responsibility given to the TCB makes it imperative that confidence in the controls it provides be established. Naturally, this applies to TCB hardware, software, and firmware. The following discussion considers only software verification. Techniques for verifying hardware correctness have tended to emphasize exhaustive testing, and will no doubt continue to do so. Even here, however, the trend is toward more formal techniques of verification, similar to those being applied to software. One approach is given in [FURT78]. IBM has done some work on microcode verification. Minimizing the complexity of TCB software is a major factor in raising the confidence level that can be assigned to the protection mechanisms it provides. Consequently, two general design goals to follow after identifying all security relevant operations for inclusion in the TCB are (a) to exclude from the TCB software any operations not strictly security-related so that one can focus attention on those that are, and (b) to make as full use as possible of protection features available in the hardware. Formal techniques of verification, such as those discussed in the next section, are promoted in TCB design to provide an acceptable methodology upon which to base a decision as to the correctness of the design and of the implementation.

TCB.3.3.3.1 Security Model

Any formal methodology for verifying the correctness of a TCB must start with the adoption of a mathematical model of

the desired protection policy. A model encompassing mandatory security and to some extent the discretionary security and integrity policies was developed by Bell and LaPadula [BELL73]. Biba [BIBA75] has shown how mandatory integrity is the dual of security and, consequently may be modeled similarly. There are five axioms of the model. The primary two are the simple security condition and the *-property (read star-property). The simple security condition states that a subject cannot observe an object unless the security level of the subject, that is, the protection attributes, is greater than or equal to that of the object. This axiom alone might be sufficient if not for the threat of non-TCB software either accidentally or intentionally copying information into objects at lower security levels. For this reason, the *-property is included. The *-property states a subject may only modify an object if the security level of the subject is less than or equal to the security level of the object.

The simple security condition and the *-property can be circumvented within a computer system by not properly classifying the object initially or by reclassifying the object arbitrarily. To prevent this, the model includes two additional axioms: the activity axiom guarantees that all objects have a well-defined security level known to the TCB; the tranquility axiom requires the classifications of objects are not changed.

The model also defines what is called a "trusted subject" that may be privileged to violate the protection policy in some ways where the policy is too restrictive. For instance, part of the TCB might be a "trusted process" that allows a user to change the security level of information that should be declassified (e.g., has been extracted from a classified document but is itself not classified). This action would normally be considered a tranquility or *-property violation, depending on whether the object containing the information had its security level changed or the information was copied into an object at a lower security level.

TCB.3.3.3.2 Methodology

A verification methodology is depicted in figure 3-3. In this technique, the correspondence between the implementation (here shown as the machine code) and protection policy is proven in three steps: (a) the properties of a mathematical model of the protection policy are proven to be upheld in a formal top level specification of the behavior of a given TCB in terms of its input, output, and side effects; (b) the implementation of the specifications in a verifiable programming language (languages such as Pascal, Gypsy, Modula, and Euclid for

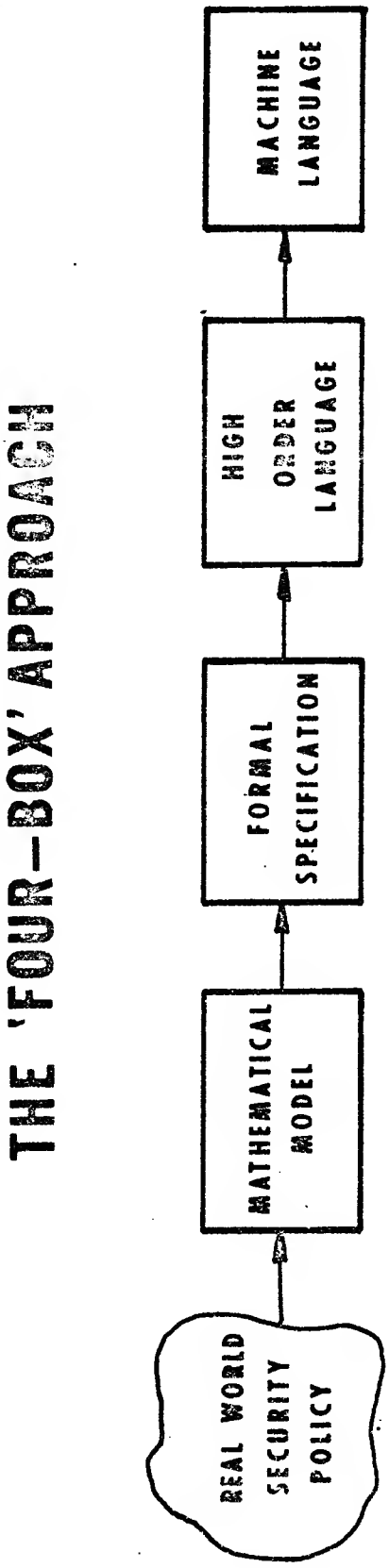

FIGURE 3-3

which verification tools either exist or are currently being planned [GOOD78b]) is shown to faithfully correspond to the formal specifications; and finally (c) the generated machine code is demonstrated to correctly implement the programs. The model describes the conditions under which the subjects in the system access the objects. With this approach, it can be shown that the machine code realizes the goals of the model, and as a result, that the specified protection is provided.

Where trusted subjects are part of the system, a similar correspondence proof starting with an additional model of the way in which the trusted subject is allowed to violate the general model becomes necessary. Clearly, the more extensive the duties of the trusted subject, the more complex the model and proof.

TCB.3.3.3.3 Confinement Problems

The TCB is designed to "confine" what a process can access in a computer system. The discussion above centers around direct access to information. Other methods exist to compromise information that are not always as easily detected or corrected. Known as "indirect channels", they exist as a side-effect of resource-sharing. This manner of passing information may be divided into "storage" channels and "timing" channels. Storage channels involve shared control variables that can be influenced by a sender and read by a receiver, for instance when the fact that the system disk is full is returned to a process trying to create a file. Storage channels, however, can be detected using verification techniques. Timing channels also involve the use of resources, but here the exchange medium is time; these channels are not easily detected through verification. An example of a timing channel is where modulation of scheduling time can be used to pass information.

In order to take advantage of indirect channels, at least two "colluding" processes are needed, one with direct access to the information desired, and a second one to detect the modulations and translate them into information that can be used by an unauthorized recipient. Such a channel might be slowed by introducing noise, for instance by varying the length of time certain operations take to complete, but performance would be affected.

Storage channels are related to the visibility of control information: data "about" information, for example, the names of files not themselves directly accessible, the length of an IPC message to another user, the time an object was last modified, or the access control list of a file. It is often the case that even the fact that an object with certain protection attributes exists is information that must be protected. Even the name of a newly created object

such as a file can be a channel if this name is dependent on information about other files, e.g., if the name is derived from an incremental counter, used only to generate new file names. This type of channel can often be closed by making the data about legitimate information as protected as the information itself. However, this is not always desirable: for instance, in computer networks, software concerned only with the transmission of messages, not with their contents, might need to view message headers containing message length, destination, etc.

Systems designers should be aware of confinement problems and the threats they pose. Formal techniques to at least identify and determine the bandwidth of the channels, if not completely close them, are certainly of value here. Ad hoc measures may be necessary in their absence.

TCB.3.4 Performance Requirements

Since the functions of the TCB are interpretive in nature, they may be slow to execute unless adequate support is provided in the hardware. For this reason, in the examples of functions given below, hardware implementations (including firmware/microcode), as opposed to software, are stressed, with the idea that reasonable performance is only accomplished when support for the protection mechanisms exists in hardware. Certainly, software implementations are not excluded, and due to the malleability of software, are likely more susceptible to appreciable optimization.

TCB.4 Detailed Requirements

The kinds of functions that would be performed by a TCB are outlined below. Those listed are general in nature: they are intended to support both general-purpose operating systems and a variety of dedicated applications that due to potential size and complexity, could not easily be verified.

The functions can be divided into two general areas: software interface functions, operations invoked by programs, and user interface functions, operations invoked directly by users. In terms of a security kernel implementation, the software interface functions would for the most part be implemented by the kernel; the user interface functions would likely be carried out in trusted processes.

TCB.4.1 Software Interface Functions

The TCB acts very much like a primitive operating system. The software interface functions are those system calls that user and application programs running in processes on top of the TCB may directly invoke. These functions fall into three categories: processes, input/output, and storage.

In the descriptions that follow, general input, output, and processing requirements are stated. Output values to processes in particular could cause confinement problems (i.e., serve as indirect channels), by relating the status of control variables that are affected by operations by other processes. Likely instances of this are mentioned wherever possible.

TCB.4.1.1 Processes

Processes are the primary active elements in the system, embodying the notion of the subject in the mathematical model. (Processes also behave as objects when communicating with each other.) By definition, a process is "an address space, a point of execution, and a unit of scheduling." More precisely, a process consists of code and data accessible as part of its address space; a program location at which at any point during the life of the process the address of the currently executing instruction can be found; and periodic access to the processor in order to continue. The role of the TCB is to manage the individual address spaces by providing a unique environment for each process, often called a "per-process virtual space", and to equitably schedule the processor among the processes. Also, since many applications require cooperating processes, an inter-process communication (IPC) mechanism is required as part of the TCB.

TCB.4.1.1.1 Create Process

A create process function causes a new per-process virtual space to be established with specific program code and an identified starting execution point. The identity of the user causing the process to be created should be associated with the process, and depending on the protection policy in force, protection attributes should be assigned, such as a security level at which the process should execute in the case of mandatory security.

TCB.4.1.1.2 Delete Process

A delete process function causes a process to be purged from the system, and its virtual space freed. The process is no longer considered a valid subject or object. If one process may delete another with different protection attributes, an indirect channel may arise from returning the fact of the success or failure of the operation to the requesting process.

TCB.4.1.1.3 Swap Process

A swap process function allows a process to become blocked and consequently enable others to run. A TCB implementation may choose to regularly schedule other processes to execute after some fixed "time-slice" has elapsed for the running process. If a TCB supports time-slicing, a swap function may not be necessary. In order to address a denial of service threat, this will not be the only process blocking operation: certain I/O operations should cause the process initiating the operation to be suspended until the operation completes.

For example, the hardware could support such an operation through mechanisms that effect fast process swaps with the corresponding change in address spaces. An example of such support is a single "descriptor base" register that points to descriptors for a process' address space, only modifiable from the privileged domain. The swap would be executed in little more than the time required for a single "move" operation.

As was mentioned above, the "scheduling" operation in itself may contribute to a timing channel, that must be carefully monitored.

TCB.4.1.1.4 IPC Send

A process may send a message to another process permitted to receive messages from it through an IPC send mechanism. The TCB should be guided by the applicable protection policy in determining whether the message should be sent, based on the protection attributes of the sending and receiving process. The TCB should also insure that messages are sent to the correct destination.

An indirect channel may result from returning the success or failure of "queuing" the message to the sending process, because the returned value may indicate the existence of other messages for the destination process, as well as the existence of the destination process. This may be a problem particularly where processes with different protection attributes are involved (even if the attributes are sufficient for actually sending the message). If such a channel is of concern, a better option might be to only return errors involving the message itself (e.g., message too long, bad message format). Clearly, there is a tradeoff here between utility and security.

TCB.4.1.1.5 IPC Receive

A process may receive a message previously sent to it through an IPC receive function. The TCB must insure that

in allowing a process to receive the message, the process does not violate the applicable protection policy.

TCB.4.1.2 Input/Output

Depending on the sophistication of the TCB, I/O operations may range from forcing the user to take care of low level control all the way to hiding from the user all device dependencies, essentially by presenting I/O devices as simple storage objects, such as described below. Where I/O details cannot be entirely hidden from the user, one could classify I/O devices as devices that can only manipulate data objects with a common protection attribute at one time (such as a line printer), and those that can manage data objects representing many different protection attributes simultaneously (such as disk storage devices). These two categories can be even further broken down into devices that can read or write any location in memory and those that can only access specific areas. These categories present special threats, but in all cases the completeness criteria must apply, requiring that the TCB mediate the movement of data from one place to another, that is, from one object to another. To resolve this problem, all I/O operations should be mediated by the TCB.

Some computer architectures only allow software running in the most privileged mode to execute instructions directing I/O. As a result, if only the TCB can assume privileged mode, TCB mediation of I/O is more easily implemented.

In the first category, if access to the device can be controlled merely by restricting access to the memory object which the device uses, the problem becomes how to properly assign the associated memory to a user's process, and no special TCB I/O functions are necessary. However, if special timing requirements must be met to adequately complete an I/O operation, quick response times may only be possible by having the TCB service the device, in which case a special operation is still needed.

When the device can contain objects having different protection attributes, the entire I/O operation will involve not only a memory object, but also a particular object on the device having the requisite protection attributes. TCB mediation in such a case is discussed under "Storage Objects."

TCB.4.1.2.1 Access Device

The access device function is a directive to the TCB to perform an I/O operation on a given device with specified data. The operations performed will depend on the device: terminals will require read and write operations at a

minimum. The TCB would determine if the protection attributes of the requesting process allow it to reference the device in the manner requested.

This kind of operation will only be necessary when mapped I/O is not possible.

TCB.4.1.2.2 Map Device

The map device operation makes the memory and control associated with a device correspond to an area in the process' address space. As in the case of the "access device" function, a process must have protection attributes commensurate to that of the information allowed on the device to successfully execute this operation. This operation may not be possible if mapped I/O is not available in the hardware.

TCB.4.1.2.3 Unmap Device

The unmap device frees a device mapped in the address space of a process.

TCB.4.1.3 Storage Objects

The term "storage objects" refers to the various logical storage areas into which data is read and written, that is, areas that are recognized as objects by the TCB. Such objects may take the form of logical files or merely recognizable units of a file such as a fixed-length block. These objects may ultimately reside on a long-term storage device, or only exist during the lifetime of the process, as required. Where long-term devices have information with varied protection attributes, as discussed in the previous section, TCB mediation results in virtualizing the device into recognizable objects each of which may take on different protection attributes. The operations on storage objects include creation, deletion, and the direct access involved in reading and writing.

TCB.4.1.3.1 Create Object

The create object function allocates a new storage object. Physical space may or may not be allocated, but if so, the amount of space actually allocated may be a system default value or specified at the time of creation.

As mentioned above, naming conventions for storage objects such as files may open an undesirable indirect channel. If the names are (unambiguously) user-defined or randomly generated by the TCB, the channel can be reduced.

TCB.4.1.3.2 Delete Object

The delete object function removes an object from the system and expunges the information and any space associated with it. The TCB first must verify that the protection attributes of the process and object allow the object to be deleted. Indirect channels in this case are similar to those for "delete process." The fact of the success or failure of the operation may cause undesirable information leakage.

TCB.4.1.3.3 Fetch Object

The fetch object function makes any data written in the object available to the calling process. The TCB must determine first if the protection attributes of the object allow it to be accessed by the process. This function may be implemented primarily in hardware, by mapping the physical address of the object into a virtual address of the caller, or in software by copying the data in the object into a region of the caller's address space.

TCB.4.1.3.4 Store Object

The store object function removes the object from the active environment of the calling process. If the object is mapped into the caller's virtual space, this function will include an unmap.

TCB.4.1.3.5 Change Object Protection Attributes

A protection policy may dictate that subjects may change some or all of the protection attributes of objects they can access. Alternatively, only trusted subjects might be allowed to change certain attributes. The TCB should determine if such a change is permitted within the limits of the protection policy.

TCB.4.2 User Interface Functions

The TCB software interface functions address the operations executable by arbitrary user or applications software. The user interface functions, on the other hand, include those operations that should be directly invokable by users. By localizing the security-critical functions in a TCB for verification, it becomes unnecessary for the remaining software running in the system to be verified before the system can be trusted to enforce a protection policy. Most applications software should be able to run securely, by merely taking advantage of TCB software interface facilities. Applications may enforce their own protection requirements in addition to those of the TCB, e.g., a data base management system may require very small files be

controlled, where the granularity of the files is too small to be feasibly protected by the TCB. In such a case, the application would still rely on the basic protection environment provided by the TCB. When users need capabilities beyond that normally provided to general applications, such as the ability to change the owner of a file object, direct contact with the TCB is required.

In kernel-based systems, the user interface functions are commonly implemented as trusted processes. Moreover, these trusted processes rely on the equivalent of the software interface functions for support.

These functions fall into three categories: user services, operations and maintenance, and administration.

TCB.4.2.1 User Services

Certain operations may be available to users as part of standard set of functions a user may wish to perform. Three are of interest here: authentication of the user to the system and of the system to the user, modification of protection attributes, and special I/O.

TCB.4.2.1.1 Authentication

The act of "logging in", of identifying oneself to the system and confirming that the system is ready to act on the behalf of the requester, is critical to the protection mechanisms, since all operations and data accesses that subsequently occur will be done in the name of this user. Consequently, identification and authentication mechanisms that play a part in validating a user to the system should be carefully designed and implemented as part of the TCB.

Likewise, the system must have some way of alerting the user when the TCB is in command of terminal communications, rather than untrusted software merely mimicking the TCB. For example, the TCB might signal to the user in a way that non-TCB software could not, or a special terminal button could be reserved for users to force the attention of the TCB, to the exclusion of all other processes.

TCB.4.2.1.2 Access Modification

Access modification functions allow a user to securely redefine the protection attributes of objects he/she controls, particularly in the case of discretionary policy. Also included here are operations that allow a user to select the protection attributes to be assumed while using the system, where the attributes may take on a range of values. For example, a user with a security level of Top Secret, may choose temporarily to operate as if Unclassified

in order to update bowling scores.

Many factors must be considered in implementing such an operation, particularly if implemented in a process. The user must have some way of convincing himself that the object for which the protection attributes are being changed is indeed what is intended. For instance, the user might be allowed to view a file to confirm its contents before changing its security level. Another issue involves the synchronization problem resulting from other processes possibly accessing the object at the instant the access modification is attempted. The TCB should prevent such a change from occurring unless the object were "locked", or temporarily made inaccessible to other processes, until the operation was complete, and also access to the other processes should be re-evaluated on completion.

TCB.4.2.1.3 Special I/O

I/O functions not covered in the software interface functions due to their specialized nature are: (a) network communications, and (b) spooling, e.g. to a line printer or mailer. The ramifications of both of these areas are too extensive to adequately cover here. The reader is referred to [KSOS78].

TCB.4.2.2 Operations/Maintenance

In the operations and maintenance category fall those functions that would normally be performed by special users, the system operators, in running and maintaining the system. Examples of such operations are system startup and shutdown, backup and restore of long-term storage, system-wide diagnostics, and system generation.

TCB.4.2.2.1 Startup/Shutdown

The security model discussed above assumes that in a TCB, an initial secure state is attained and that subsequent operations on the system obey the protection policy and do not affect the security of the system. This characteristic of a TCB can be said to be true regardless of the protection policy and security model employed. A "startup", or bootstrap, operation addresses the initialization of the system and the establishment of the protection environment upon which subsequent operations are based. The model itself, or the formal specifications of a specific design, can address what the characteristics of all secure states are, and hence the requirements for the initial secure state. Consequently, programs that create this state can be well-defined. Since it is the operator who must execute the necessary procedures that initialize the system, TCB functions interfacing the operator must be trusted to do

what the operator specifies.

Shutdown procedures are equally crucial in that an arbitrary suspension of system activities could easily leave the system in an incomplete state, making it difficult to resume securely (for instance, if only half of an updated password file is moved back to disk). One must, for instance, write all memory-resident tables out to disk where necessary.

TCB.4.2.2.2 Backup/Restore

To allow for recovery from unpredictable hardware failure, and consequently the arbitrary suspension mentioned above, "checkpoints" may be taken of a given state of the storage system, for instance, by copying all files from disk to some other medium, such as magnetic tape. In the event of system failure, the state of files at some earlier time can be recovered. The backup function must operate on the system in a consistent state, and accurately reflect that state; the restore function must reliably rebuild from the last completely consistent record it has of a secure state. Note that the backup system requires an especially high level of trust since it stores protection attributes as well as data.

TCB.4.2.2.3 Diagnostics

Diagnostics of both hardware and software integrity can thwart potentially harmful situations. In particular, hardware diagnostics attempt to signal when problems arise, or, when something has already gone wrong, they try to aid the technician in pinpointing where the problem is. Diagnostics written in software typically access all areas of memory and devices, and consequently, if run during normal operation of the rest of the system, require tight TCB controls. If possible, they should be relegated to user programs and limited to specific access spaces during the course of their operation. However, in such a case it would be impossible to test the security critical hardware, such as descriptor registers if present. Such software, for on-line diagnosis, must be included in the TCB, and limited to operator use.

TCB.4.2.2.4 System Generation

System generation deals with creating the program modules in executable form that can subsequently be loaded during system startup. It is included here for completeness, although there is no intention to require that editors, compilers, loaders, and so forth, be verified to correctly produce the code that is later verified correct. Correct system generation is an area that is clearly vulnerable, and procedures must be made to ensure that the master source is not intentionally corrupted.

TCB.4.2.3 Administration

The administration and overall management of a system both in terms of daily operations and security operations may be relegated to a user, or users, other than the system operator. Functions in support of system administration include but are not limited to updating data bases of users and their valid protection attributes; and audit and surveillance of protection violations;

TCB.4.2.3.1 User Data Base Updates

A typical user data base would contain at a minimum the names of valid users, their authentication data (e.g., password, voice print, fingerprints), and information relating to the protection attributes each user may take on while using the system. TCB functions must be available to an administrator to allow updates to the data base in such a way that the new information is faithfully represented to the user authentication mechanism.

TCB.4.2.3.2 Audit and Surveillance

Audit facilities capture and securely record significant events in the system, including potential protection violations, and provide functions to access and review the data. Surveillance facilities allow for real-time inspection of system activities. Audit and surveillance mechanisms provide an additional layer of protection. They should be implemented as part of a TCB not only because they require access to all activities on the system as they occur, but also since if they are not themselves verified to be correct and complete, flagrant violations might go undetected.

(End of the TCB extract.)

3.3 TECHNOLOGY TRANSFER ACTIVITIES

Once the requirements have been defined, a second significant prerequisite to achieving the widespread availability of commercial trusted systems is the transfer of the computer security technology, developed largely under the auspices of the DoD, to industry. This technology, although available (in part) in the open literature, had never been presented in a cohesive and detailed manner. To stimulate technology transfer, the Initiative sponsors a series of computer security seminars aimed at Consortium members, the computer industry, and general computer users. Consortium members are also actively involved in nation-wide conferences and workshops addressing computer security and computer systems in general. Descriptions of some of these activities follow in chronological order.

3.3.1 The MITRE Computer Security Workshop

During the week of 29 January to 2 February 1979, MITRE Corporation personnel conducted a computer security workshop for DoD personnel. The workshop involved eight general lectures, five different technical workshop groups, and four guest lecturers.

The goal of the workshop was to bring together for the first time all of the technology, background, and experience necessary for a DoD program manager to understand the state-of-the-art in computer internal security controls (e.g. the security kernel), but the material included traditional concepts (e.g. periods processing with color changes) as well.

There were 53 registered attendees from DoD and related agencies, including one person from the Canadian Department of National Defense. Among the agencies and services represented were: NSA, DCA, DIA, WSEO, DoDCI, USAF, ESD, RADC, SAC, DSDC, DA, DARCOM, NESC, NOSC, USMC, and CINCPAC.

The general lectures were presented on the following topics:

Introduction, History, and Background
Operating systems and Security Kernel Organization
Mathematics of Secure Systems
Specification of Verifiably-Secure systems
Secure Computer System Developments (KSOS, SCOMP, KVM/370)
Design and Verification fo Secure systems
Secure systems: Experience, Certification, and Procurement
Secure systems: Present and Future

The guest lecturers were:

Mr. Stephen T. Walker
Staff Assistant, OUSDRE/C3I
"DOD Computer Security Initiative"

Mr. Steven B. Lipner
ADH, MITRE Corporation
"The Evolution of Computer Security Technology"

Mr. Clark Weissman
Chief Technologist, System Development Corporation
"System Security Analysis/Certification: Methodology and Results"

Prof. Jerome Saltzer
Professor of Computer Science and Engineering, MIT
"Security and Changing Technology"

Technical Workshops were given on the following topics:

Basic Principles
Security Kernel/Non-Kernel Security-Related Software Design
Secure system Verification
Secure Computer Environment
Capability Architecture

3.3.2 1979 National Computer Conference

On June 4-7, 1979, the 1979 National Computer Conference was held. An entire session of the conference was devoted to the Initiative. Seven technical papers were prepared for the session, which was chaired by Mr. Stephen T. Walker, OUSDRE/C3I. The papers appear in the proceedings of the conference. The papers, and their authors, are as follows:

"Applications for Multilevel Secure Operating Systems," John P. L. Woodward, The MITRE Corporation.

"The Foundations of a Provably Secure Operating System (PSOS)", Richard J. Feiertag and Peter G. Neumann, SRI International.

"A Security Retrofit of VM/370," B. D. Gold, R. R. Linde, R. J. Peeler, M. Schaefer, J. F. Scheid, and P. D. Ward, System Development Corporation.

"KSOS - The Design of a Secure Operating System," E. J. McCauley and P. J. Drongowski, Ford Aerospace and Communications Corporation.

"UCLA Secure UNIX," Gerald J. Popek, Mark Kampe, Charles S. Kline, Allen Stoughton, Michael Urban, and Evelyn J. Walton, UCLA.

"KSOS - Development Methodology for a Secure Operating System," T. A. Berson and G. L. Barksdale, Jr., Ford Aerospace and Communications Corporation.

"KSOS - Computer Network Applications," M. A. Padlipsky, K. J. Biba, and R. B. Neely, Ford Aerospace and Communications Corporation.

3.3.3 1979 Summer Study on Air Force Computer Security

During June and July 1979, the Air Force Office of Scientific Research sponsored a summer study on Air Force computer security issues. The Initiative provided extensive support and assistance to the study. Following is a significant portion of the Executive Summary written by Dr. J. Barton DeWolf and Paul A. Szulewski, editors of the study report [DEW079].

The study was held at the Charles Stark Draper Laboratory, Inc. (CSDL), with some sessions at Hanscom Air Force Base in Bedford, MA and at the MITRE Corporation in Bedford, MA. The objectives of the study were to evaluate current research and development in relation to Air Force requirements for multilevel secure computer systems, to identify critical research issues, and to provide guidance and recommendations for future research and development emphasis. To this end, over 150 attendees representing academic, industrial, civilian government, and military organizations, participated from June 18 through July 13 in an intensive technology review and evaluation.

The summer study was divided into the following nine sessions, each lasting from 1 to 3 days.

(1) Air Force Computer Security Requirements.

(2) Security Working Group Meeting.

(3) Trusted Operating Systems.

(4) Verification Technology.

(5) Secure Data-Base Management.

(6) Secure Systems Evaluation.

(7) Secure Distributed Systems and Applications.

(8) Air Force Computer Security Policy.

(9) Summary and Research Recommendations.

Although all the sessions shared a common format, each individual session chairperson was responsible for the specific form and content of his or her session. Participants, in general, prepared only slides to supplement their oral presentations. Typically, each session began with short presentations by each of the participants, which served to provide an overview of the technology and to stimulate ideas and discussion. The presentations were followed by discussion periods, in which questions of interest were addressed. Certain participants knowledgeable in the pertinent areas of computer security under discussion were selected to summarize the sessions in detail. These session summaries form the body of this report. The remainder of this executive summary highlights key findings and recommendations.

In the keynote presentation on the opening day, Major General Robert Herres described the multilevel security problem as a dilemma:

> "...on the one hand, we must maintain the security and integrity of our sensitive information, but on the other hand, we must be able to respond quickly to rapidly changing situations, especially during times of crisis or war. And this means we must process and distribute information rapidly among many people at different levels of command, and possessing a variety of clearances and 'needs to know'.
>
> "We cannot let security considerations throttle our operational responsiveness, but we also cannot jeopardize sources of intelligence information, war plans, actions or sensitive information by having some unknown hole in our security which could be exploited by some individual or group, quite undetectably."

The Requirements Session emphasized the need for solutions to problems arising from the sharing of sensitive information in computer systems. Presentations were made by representatives of the Defense Communications Agency (WWMCCS program), ESD (OASIS program), the Military Airlift Command, the Air Force Weapons Laboratory, the Defense Mapping Agency, and the Rome Air Development Center (KAIS program; other tactical programs). In general, it was found that requirements had not changed significantly from those reported in the 1972 study, but that the trend towards distributed processing and computer networks was adding a

new dimension of urgency and complexity to the problem. The presentations described current modes of processing classified information as combinations of dedicated, system high, and periods processing. These modes entail numerous inefficiencies which include the following.

(1) Waste of computer resources.

(2) Overclassification of information.

(3) Overclearing of users.

(4) Excessive number of clearances.

(5) Duplication of hardware and software.

(6) Reliance on cumbersome, costly, and time-consuming manual procedures for review and declassification of information.

There was also widespread concern regarding the cost of converting or adapting existing software and data for use with new hardware or operating systems, though efficiency gains resulting from the use of multilevel secure systems would tend to offset the conversion costs. The impact (including the cost impact) of computer security requirements on the accomplishment of Air Force mission objectives has not been fully analyzed.

The Working Group Session discussed topics which were covered in greater detail in the other sessions; therefore, a separate summary is not included herein.

The Trusted Operating Systems Session brought together a panel of 12 practitioners--persons actively involved in the design and development of trusted systems--to discuss their experiences and views on system architecture, hardware support, and development methodologies. Most recent trusted system development activity has followed the kernelized operating system approach recommended by the 1972 ESD planning study. In this approach, software specifications for the security management portion of the operating system (i.e., the kernel) are proven to be in conformance with a mathematical model of the security policy. This approach has been successful in producing several prototype implementations of trusted operating systems, with a number of production versions nearing completion. However, opportunities to develop applications programs on these systems have been very limited, and experience is badly needed. In the past, operating system penetration studies have been useful in demonstrating protection mechanism weaknesses, and future studies will be needed on the new generation of trusted systems. In general, panel members

felt that such studies would show them to be far more secure than their predecessors. With respect to hardware support for trusted systems, the panel felt that, although the situation has been improving, several areas were in need of research emphasis. These included the following.

(1) Hardware-mapped input/output (I/O).

(2) Protection of logical objects.

(3) Unified device interfaces.

(4) Multiple domains (more than two).

(5) Fast domain switching.

The Verification Technology Session served to emphasize the essential role that formal specification and verification have played in the development of trusted systems. As mentioned previously, formal verification or proof techniques have been used to show the correspondence between the kernel specifications and the mathematical security model. Current specification and verification approaches and tools are limited in capability, however; and (for the most part) have not been used to show the correspondence between the code and the specifications. Furthermore, current verification systems are usable only by a small community of educated designers; and there is a need both to make the tools easier to use and to enlarge the user community. Despite these limitations, verification technology has matured to the point where it is desirable to attempt verification through the code level on limited-scale real applications, such as clear-core routines and labeling utilities. It is also desirable to develop methods to verify that firmware and hardware have been implemented in accordance with their specifications. One of the highlights of this session was an on-line demonstration of the Gypsy and AFFIRM verification systems.

The Secure Data-Base Management (DBM) Session dealt with a challenging applications area in need of future research and development emphasis. Security technology for the DBM problem is still in its infancy. To date, the limited experience in the application of trusted operating system technology to DBM issues suggests that several problems need attention. A critical issue is whether current mathematical security models are adequate for multilevel data bases. Data-base constructs not well addressed in current models include multi-level objects and multilevel relations, aggregation and inference, and data objects--the sensitivity of which is content-dependent. Also the support provided in some trusted operating system designs may not be adequate for DBM applications. To be useful for DBM, the operating

system should support access control on finer granularity data objects than most current systems support (e.g., files). The user interface to the data base is another area of concern.

The Secure System Evaluation Session addressed the need to establish a DoD secure system approval process--a critical element of the computer security initiative recently undertaken within the DoD. The session focused on

(1) The technical evaluation and categorization of trusted systems.

(2) The characteristics of threat environments and applications.

Seven levels of protection were proposed for evaluating trusted systems. The threat environment was characterized in terms of processor coupling, user/data exposure, developer/user trust, and user capability. The session provided evidence that a workable evaluation process could be established, and that a consensus could be reached matching threat environments with a desired level of protection. A key assumption throughout the session was that limiting the user's capability (e.g., use of function keys, transaction processing in a nonprogramming environment) significantly reduces the security risk. Since the security requirements of such systems are not well understood, this is an area recommended for future research.

The Secure Distributed Systems and Applications Session discussed approaches to providing multilevel secure computer network services. The presentations included discussion of SACDIN, the KSOS network connection, the military message experiment, and several other systems. It appears that the trend towards distributed systems can benefit system effectiveness, but it exposes information to additional security threats such as integrity violations, indirect communication channels, and incorrect user authentication. Some approaches are emerging for the use of encryption in secure networks, but more work is needed in this area. Key areas for future research include the following.

(1) Design methodologies.

(2) Policy issues.

(3) Communications protocols for multilevel secure networks.

The Air Force Computer Security Policy Session dealt with current DoD policy as set forth in DODD 5200.28, as implemented in Air Force Regulation 300-8. Current computer security

policy often inhibits the operational capability of Automatic Data Processing (ADP) systems, as was emphasized during the session on requirements. The problem will be alleviated as multilevel secure computing systems become widely available. The situation would also be improved if current policy more adequately took into account the degree of risk in various operational environments. Low-risk environments could then utilize less costly rules and procedures. As was pointed out on several occasions during the summer study, current policy needs to be extended to cover other data-processing issues: fraud, privacy, data integrity, declassification, aggregation, sanitization, and denial of service. An informal statement of current policy on these issues would assist the development of formal mathematical models.

To summarize, in the last few years, the field of computer security has made significant progress towards the goal of trusted computing systems for multilevel-secure applications. The following research and development goals were generated by the group in the final session.

(1) Continued support for ongoing trusted operating system projects (e.g., KSOS, KVM/370).

(2) Increased support for future applications to be hosted on these systems.

(3) Research to improve hardware support for trusted operating systems with emphasis on hardware-mapped I/O, protection of logical objects, unified device interfaces, multiple domains, and fast domain switching.

(4) Verification-methodology research with a focus on practical, real applications of limited scale.

(5) Research to improve the hardware support for verification and to improve verification system support tools.

(6) Research to develop methods to verify that hardware and firmware have been implemented in accordance with their specifications.

(7) Research to identify trusted operating system enhancements needed to support data-base management applications.

(8) Improved technology transfer between academic, industrial, civilian government, and military domains in the computer security field.

(9) Standardization of terms for the ADP security community.

(10) Research to define the security requirements of limited-capability systems.

(11) Research to concentrate on design methodologies, policy issues, and communications protocols for trusted distributed processing architectures.

(12) Development of approaches to detect and control indirect communication channels (timing channels).

(13) Continued research on encryption approaches and their relation to kernel technology and capability architectures.

(14) Research to extend formal (mathematical) policy models to cover the problems of fraud, privacy, multilevel data bases, data integrity, declassification, aggregation, sanitization, and denial of service.

(15) Development of methods to evaluate security risk in ADP systems in terms of threat identification and quantification of loss.

The Air Force needs multilevel secure systems. The technology is at hand. An active and ongoing research and development program is needed to make the technology widely available and useful over a broad range of applications.

(End of Summer Study executive summary.)

3.3.4 July 1979 Industry Seminar

On 17 and 18 July 1979, the Initiative conducted its first industry technical seminar at the National Bureau of Standards in Gaithersburg, Maryland. The 280 attendees were drawn almost equally from computer manufacturers, system houses, and government agencies. The objective of this seminar was to acquaint computer system developers and users with the status of the development of "trusted" ADP systems with the DOD and the current planning for the evaluation of the integrity of commercial implementations of these systems. The seminar presented an overview of a number of topics essential to the development of "trusted" ADP systems. Much of the material presented was of a technical nature intended for computer system designers and software system engineers. However, the sophisticated computer user in the Federal government and in private industry should have found the seminar useful in understanding security characteristics of future systems.

PROGRAM

SEMINAR ON
DEPARTMENT OF DEFENSE

COMPUTER SECURITY INITIATIVE

... to achieve the widespread availability
of trusted computer systems

July 17, 1979

8:30 am	Registration at National Bureau of Standards	
9:15	Opening Remarks	- James H. Burrows, Director Institute for Computer Sciences and Technology National Bureau of Standards
9:30	Keynote Address	- "Computer Security Requirements in the DoD" Honorable Gerald P. Dinneen Assistant Secretary of Defense for Communications, Command, Control and Intelligence
10:00		- "Computer Security Requirements Beyond the DoD" Dr. Willis Ware Rand Corporation
10:30	Coffee Break	
10:45		- DoD Computer Security Initiative Program Background and Perspective Stephen T. Walker Chairman, DoD Computer Security Technical Consortium
11:30		- Protection of Operating Systems Edmund Burke MITRE Corporation
1:00 pm	Lunch	
2:00		- Kernel Design Methodology LtCol Roger Schell, USAF Naval Post Graduate School
3:15	Break	
3:30		- Formal Specification and Verification Peter Tasker MITRE Corporation
4:30	Adjourn	

FIGURE 3-4

July 18, 1979

9:00 am		- Secure System Developments Kernelized Secure Operating System (KSOS) Dr. E. J. McCauley Ford Aerospace and Communications Corporation Kernelized VM-370 Operating System (KVM) Marvin Schaefer System Development Corporation
11:00	Coffee Break	
11:15		- Secure Communications Processor Matti Kert Honeywell Corporation
12:00		- Secure System Applications John P. L. Woodward MITRE Corporation
1:00 pm	Lunch	
2:00		- DoD Computer Security Initiative Stephen T. Walker
3:30	Adjourn	

FIGURE 3-4 CONCLUDED

Figure 3-4 shows the program for the seminar.

3.3.5 January 1980 Industry Seminar

On 15-17 January 1980, the second Initiative-sponsored industry seminar was held at the National Bureau of Standards. This seminar was a key part of the task to transfer the computer security technology to industry (especially the computer manufacturers) and to those who will be using and buying trusted computers. There were 300 attendees from industry and government along with about 30 participants.

The seminar was organized into three sessions: a general introductory and keynote session on 15 January; a policy and requirements session on 16 and 17 January; and a parallel session on Trusted Computing Base (TCB) design on 16 and 17 January. The first of the two parallel sessions provided in depth discussions of policy issues as they apply to multilevel secure computer systems, an analysis of applications of such systems within the DoD and beyond, and a presentation of the Trusted Computing Base concept. The TCB session, intended for operating system developers and sophisticated computer science technical experts, provided a detailed analysis of the Trusted Computing Base concept, which is the emerging generalized basis upon which high integrity operating systems may be evaluated, followed by discussions by the principle designers of the major DoD trusted system developments relating their systems to the TCB concept.

Figure 3-5 is a copy of the seminar program.

3.3.6 November 1980 Industry Seminar

On 18-20 November 1980, the Initiative conducted its third industry seminar at the National Bureau of Standards. This was the latest in the series of seminars to acquaint computer system developers and users with the status of trusted ADP system developments and evaluation. There were 380 people registered for the seminar.

The first day of the seminar included an update on the status of the Initiative and presentations by five computer manufacturers on the trusted system development activities within their organizations. Following these presentations was a panel discussion on "How can the government and the computer industry solve the computer security problem?"

The second day of the seminar opened with a discussion of the technical evaluation criteria that have been proposed as a basis for determining the relative merits of computer systems. The discussion of the assurance aspects of those

PROGRAM

Second Seminar on the Department of Defense Computer Security Initiative

National Bureau of Standards
Gaithersburg, Maryland

January 15, 1980 — Red Auditorium

9:30 am — "The Impact of Computer Security in the Intelligence Community"

Dr. John Koehler
Deputy Director for Central Intelligence for Resource Management

"The Impact of Computer Security in the Department of Defense"

Dr. Irwin Lebow
Chief Scientist
Defense Communications Agency

"The Impact of Computer Security in the Federal Government"

Mr. James Burrows
Director, Institute for Computer Science and Technology
National Bureau of Standards

BREAK

"The Impact of Computer Security in the Private Sector"

Mr. Ed Jacks
General Motors Corporation

"Status of the DoD Computer Security Initiative"

Mr. Stephen T. Walker
Chairman, DoD Computer Security Technical Consortium

1:00 pm — LUNCH

FIGURE 3-5

January 15, 1980
(Continued)

2:00 pm — "Computer Security Impacts on Near Term Systems"

Mr. Clark Weissman
System Development Corporation

"Computer Security Impacts on Future System Architectures"

Mr. Ed Burke
MITRE Corporation

BREAK

A "discussion" of what the computer manufacturers would like/should expect to hear from government users about trusted computer systems

Dr. Theodore M.P. Lee
UNIVAC Corporation

Mr. James P. Anderson
James P. Anderson Company

4:30 pm — ADJOURN

January 16-17, 1980 — TWO PARALLEL SESSIONS

SESSION I — Gneral Session - Red Auditorium

January 16, 1980

9:15 am — "Policy Issues Relating to Computer Security"

Session Chairman: Robert Campbell
Advanced Information Management, Inc.

Mr. Cecil Phillips
Chairman, Computer Security Subcommittee
DCI Security Committee

Mr. Eugene Epperly
Counterintelligence & Security Policy Directorate
Office of the Secretary of Defense
Pentagon

Mr. Robert Campbell
Advanced Information Management, Inc.

Mr. Philip R. Manuel
Phillip R. Manuel and Associates

Dr. Stockton Gaines
RAND Corporation

1:00 pm — LUNCH

FIGURE 3-5 CONTINUED

January 16, 1980
(Continued)

2:00 pm	"User Requirements and Applications"
	Session Chairman: Dr. Stockton Gaines RAND Corporation
	Mr. Larry Bernosky WWMCCS System Engineering Office
	LtCol Cerny Federal Republic of Germany Air Force
	BREAK
	Dr. Tom Berson SYTEK Corporation
	Mr. Mervyn Stuckey U.S. Department of Housing and Urban Development
4:00 pm	ADJOURN

January 17, 1980

SESSION I

9:15 am	"User Requirements and Applications" (continued)
	Dr. Von Der Brueck IABG, Germany
	Mr. John Rehbehn Social Security Administration
	Mr. William Nugent Library of Congress
	Mr. Howard Crumb Federal Reserve Bank of New York
	BREAK
	"Trusted Computing Base Concepts"
	Mr. Peter Tasker MITRE Corporation
1:00 pm	LUNCH
2:00 pm	GENERAL DISCUSSION and WRAPUP
	Mr. Stephen T. Walker

FIGURE 3-5 CONTINUED

PROGRAM

November 18, 1980 Red Auditorium

9:15 Opening Remarks

Seymour Jeffries,
Institute for Computer Sciences & Technology
National Bureau of Standards

DOD Computer Security Initiative

Stephen T. Walker, Chairman
DOD Computer Security Technical Consortium

INDUSTRY TRUSTED SYSTEM ACTIVITIES

Paul A. Karger
Digital Equipment Corporation

10:45 Break

11:00 INDUSTRY TRUSTED SYSTEM ACTIVITIES - Continued

Irma Wyman
Honeywell

Viktors Berstis
IBM

Jay Jonekait
TYMSHARE, Inc.

Theodore M. P. Lee
Sperry-Univac

1:00 Lunch

2:00 PANEL: "How Can the Government and the Computer Industry Solve the Computer Security Problem?"

Theodore M. P. Lee, Sperry-Univac
James P. Anderson, Consultant
William Eisner, Central Intelligence Agency
Steven P. Lipner, Mitre Corporation
Marvin Schaefer, System Development Corporation

3:00 Break

3:15 PANEL - Continued

4:30 Adjourn

FIGURE 3-6

November 19, 1980 Red Auditorium

9:00 "Quality Assurance and Evaluation Criteria"

Grace H. Nibaldi
Mitre Corporation

9:50 "Specification and Verification Overview"

William F. Wilson
Mitre Corporation

10:45 Break

SPECIFICATION AND VERIFICATION SYSTEMS

11:00 "FDM: A Formal Methodology for Software Development"

Richard Kemmerer
System Development Corporation

12:00 "Building Verified Systems with Gypsy"

Donald I. Good
University of Texas

1:00 Lunch

SPECIFICATION AND VERIFICATION SYSTEMS - Continued

2:00 "An Informal View of HDM's Computational Model"

Karl N. Levitt
SRI International

3:00 Break

3:15 "AFFIRM: A Specification and Verification System"

Susan L. Gerhart
USC Information Sciences Institute

4:15 Adjourn

FIGURE 3-6 CONTINUED

November 20, 1980 Red Auditorium

9:00 "An Overview of Software Testing"

Mary Jo Reece
Mitre Corporation

THE EXPERIENCES OF TRUSTED SYSTEM DEVELOPERS

9:45 "Update on KSOS"

John Nagle
Ford Aerospace and Communications Corporation

10:45 Break

11:00 KVM-370

Marvin Schaefer
System Development Corporation

12:00 "Kernelized Secure Operating System (KSOS-6)"

Charles H. Bonneau
Honeywell

1:00 Lunch

2:00 PANEL: "Where Would You Put Your Assurance Dollars?"

Panelists: Developers, Researchers, & Testers

3:00 Break

3:15 PANEL - Continued

4:15 Adjourn

FIGURE 3-6 CONCLUDED

3.4 TRUSTED COMPUTER SYSTEM EVALUATION

This section proposes an evaluation process by which trusted computer system developments may be reviewed and evaluated under the Initiative. The results of applying the process will be to develop a list of products that have undergone evaluation, and thus are eligible for use in applications requiring a trusted system. This list of systems has been designated an evaluated products list (see section 3.1). Trotter and Tasker have documented the proposed evaluation process [TROT80]. trusted computer systems. This section contains a condensation of that paper.

There are three prime elements to the evaluation process: the TCB provides the requirements; evaluation criteria have been proposed and are being coordinated with industry; and a plan has been advanced for a government-wide evaluation center. The TCB was described in section 3.2. The subsections below discuss the criteria and the center, and then present the proposed evaluation process.

3.4.1 Evaluation Criteria

An important requirement of an evaluation program, both from the viewpoint of the manufacturer and the government, is that the evaluation be consistent for all manufacturers and all products. To achieve this, a detailed set of evaluation criteria is needed that will allow both the protection value of architectural features and the assurance value of development and validation techniques to be considered. In addition, it is necessary that the criteria be independent of architecture so that innovation is not impeded. Three evaluation factors have been defined, and various degrees of rigor for each factor have been incorporated into seven hierarchical protection levels representing both system-wide protection and assurance that the protection is properly implemented. The evaluation criteria address two aspects of a system considered essential: completeness (is the policy adequate) and verifiability (how convincingly can the system be shown to implement the policy).

The proposed evaluation criteria are summarized here, and are documented in detail by Nibaldi [NIBA79b]. It should be emphasized that these criteria are preliminary and are undergoing review.

3.4.1.1 Factors

There are three prime evaluation factors: policy, mechanism, and assurance. These factors are shown in figure 3-7, and are briefly described below. They are fully described and developed in [NIBA79b].

- Policy
- Mechanism
  - Prevention
  - Detection
  - Recovery
- Assurance
  - Development Phases
    - Design
    - Implementation
  - Validation Phases
    - Testing
    - Verification
  - Operations/Maintenance

Figure 3-7. EVALUATION FACTORS

Policy

A protection policy specifies under what conditions information stored in the computer and computer resources might be shared, typically placing controls on the disclosure and/or modification of information. If there is a clear, concise statement (and hence, understanding) of the protection policy a trusted system purports to obey, then an evaluator of the system can better determine (through testing or other forms of validation) if the system enforces the stated policy. In fact, formal methods of design verification depend on precisely stated policy "models" to make rigorous mathematical proofs of correctness.

Mechanism

The mechanisms that actually enforce the protection policy may include both hardware and software. To be effective, they too must be complete and verifiable, but in addition, they must be self-protecting, able to maintain their effectiveness in the face of accidents or malicious attack by users or their programs. Operating systems can potentially confine users so that unauthorized access cannot occur, yet if they are poorly implemented, they have the potential to undermine even safeguards that are built into user programs or applications. As a result, an evaluation is expected to concentrate on operating system and related software and hardware controls, particularly in the areas of detection and prevention of policy breaches, recovery from errors, and system operations and maintenance. The TCB is the basis for the evaluation of the mechanism.

Assurance

The evaluation criteria should take into account not only that a system promises to provide a certain amount of protection (by having a suitable policy and exhibiting the appropriate mechanisms) but also that it can deliver that protection with some degree of confidence. Absolute certainty is beyond the state-of-the-art in software engineering, but steps taken in the design, implementation, and validation phases of a trusted system's development are known to raise the level of confidence one has in the system. These steps include, for instance, top-down design, structured programming, penetration testing, and mathematical verification of conformance to policy.

3.4.1.2 Protection Levels

Seven protection levels (six levels and null) are defined [NIBA79b]. These levels are cumulative in that a rating at a certain level requires that the criteria at that level and all lower levels be satisfied. When a system is evaluated it will receive a rating determined by the highest protection level that is completely satisfied. Thus a system that has satisfied all of the requirements except one for a "Level 3" will be assigned a "Level 2."

This criteria has been defined so that in the lowest levels, a system must first meet certain policy standards, even if its mechanisms are not deemed sufficiently strong to counter certain subtle attacks. At higher levels, the emphasis shifts to the evidence that the software, and ultimately the hardware, is correct.

LEVEL 0: NO PROTECTION

When there is no indication in any of the three areas that a system can protect information, the system receives a level 0 evaluation.

LEVEL 1: LIMITED CONTROLLED SHARING

Level 1 applies to systems in which the presence of data access controls that are capable of providing only limited protection are recognized.

LEVEL 2: EXTENSIVE MANDATORY SECURITY

The system protection provides: 1) administratively controlled authorization to read data, 2) flow control to prevent data compromise, and 3) write access control.

LEVEL 3: STRUCTURED PROTECTION MECHANISM

The protection mechanisms must be clearly identified, isolated and made independent of other software. Trust is gained through methodological design of the protection-related components of the operating system (i.e., the TCB) and modern programming techniques, adequate test results are still the primary means of assurance.

LEVEL 4: DESIGN CORRESPONDENCE

At this level formal methods are employed to confirm trustworthiness of the design. Mathematical proofs of correspondence of the design to a security model are required.

LEVEL 5: IMPLEMENTATION CORRESPONDENCE

The system must be shown to formally correspond to the verified top-level design, and more stringent requirements for denial of service provisions, hardware fault tolerance, and leakage channel control are demanded.

LEVEL 6: OBJECT CODE ANALYSIS

A formal analysis of the object code is required as final evidence that the implementation software fulfills the requirements of the security model. More rigorous scrutiny of the hardware base is demanded, and formal methods of verification must also be applied to the hardware.

3.4.2 Security Evaluation Center

It is proposed that the evaluation function be performed by a government-wide computer security evaluation center so that the evaluations will be as consistent as possible and so that scarce technical personnel can be best utilized. To properly apply the evaluation process, the center will maintain a staff experienced in security issues, TCB design, system design, testing, penetration, and interaction. In addition to the evaluation of industry secure systems, the staff will be available to government agencies requiring design or consultation on individual products or contracts, particularly in the area of design of applications software. Also, the center will establish and maintain an internal research and development capability to both enhance current and create new development tools essential to the system evaluation process.

3.4.3 Trusted Computer System Evaluation Process

The proposed evaluation process consists of four sequential steps: 1) preliminary evaluation, 2) interactive evaluation, 3) final evaluation, and 4) periodic re-evaluation. The preliminary evaluation step is a determination of the suitability of an industry developed system for evaluation based upon the design of the TCB of the system. When the TCB has been adequately specified, the system will be ready for an interactive evaluation. The interactive evaluation is a review of the system design in terms of the TCB and the means by which the system satisfies the criteria for the level of protection which the manufacturer specifies. The final evaluation involves analysis and testing of the completed system to determine the level of protection provided and the strengths and weaknesses relative to that level. This description presumes that these evaluations will be performed by a government-wide evaluation center.

3.4.3.1 Preliminary Evaluation

Preliminary evaluation is an analysis of the TCB of a manufacturer's system to determine the adequacy of that system for use in an environment requiring trusted access controls. The purpose of this evaluation is to determine whether or not the manufacturer's system is sufficiently designed and documented, in terms of the TCB and the evaluation criteria, to begin an interactive evaluation. When the manufacturer requests an evaluation, he will provide the evaluation center with complete system documentation and indicate the target level of protection he hopes to achieve. This can best be accomplished through a presentation given by the manufacturer describing the computer system under development in terms of the TCB specification, and detailing the design and implementation of the system in terms of the technical evaluation criteria. The preliminary evaluation will determine if the TCB can provide this "target" level of protection by analysis of the design methodology and the hardware and software mechanisms provided by the system.

When the security evaluation center receives a request to evaluate a system, a team will be formed to perform the evaluation. The output of the preliminary evaluation will be the team's assessment of the status of the system, and the potential the system has for achieving the level of protection stated by the manufacturer or the highest level the system might achieve based on the information available. The assessment may indicate that the system is not yet ready to proceed to a full interactive evaluation. This would occur if the specification has not been well defined in terms of the TCB, or if the complexity or method of implementing the TCB is not amenable to this type of

evaluation. In that case, the evaluation team will identify what further information is needed, or what steps should be taken before the system is ready for interactive evaluation.

Although it is presumed that most evaluations will be conducted on systems that have been designed with verifiable protection in mind from the onset, there may be released (production) systems that have a sufficient protection base and data protection capability to allow restructuring of the system to incorporate a TCB. In this case, the focus of the preliminary evaluation will be on the changes necessary to the production system (in structure, documentation and testing) to satisfy the criteria for one of the protection levels.

3.4.3.2 Interactive Evaluation

The interactive evaluation is a logical extension of the preliminary evaluation, which will begin when a preliminary evaluation indicates the product is suitable as a trusted system. The review of the system will focus on the TCB, while the review of the system design will focus on the evaluation criteria (i.e. how the design satisfies the criteria for the level of protection specified in the preliminary evaluation). The method of conducting the evaluation will be a series of presentations given by the developer, together with documentation appropriate to the level of development of the system. The areas of hardware and software which were covered in the preliminary evaluation will now be covered in depth by the manufacturer's design team. The manufacturer will determine the schedule for presentations based upon his progress in developing the system. One possible method is to tie the presentation schedule to the manufacturer's internal design review cycle.

The evaluation team will review the system design and point out security relevant design tradeoffs that may have been overlooked. In no case will the team attempt to re-design the manufacturer's system. The issue addressed is the compromise of the system through data security and integrity flaws, timing and storage channels, and denial of service. The evaluation team will provide the manufacturer with in progress reports detailing the teams assessment of the TCB design issues, and supplying feedback on the protection provided by the system. The manufacturer will not be required to supply special documentation defining the TCB provided the internal documentation adequately defines the system design. KSOS-6 (SCOMP KSOS) [BONN80a,BONN80b] and KSOS-11 [KSOS78] specifications provide examples of the type of specifications required for adequate system definition.

The interactive evaluation of an industry system will be complete when the analysis of all specifications is complete. The computer system will then be ready for final evaluation.

If the evaluation has been initiated prior product release it will occur concurrently with the development of the system. If the evaluation process is initiated later, in anticipation of subsequent releases of "trusted versions" of the system, any interactive evaluation step would take place during the manufacturer's formulation of the releases.

3.4.3.3 Final Evaluation

The final evaluation consists of analysis and testing of the production system to determine its strengths and weaknesses relative to the mechanisms provided for the level of protection which was originally specified. The developers will provide the evaluation center with a production system, or suitable access to one, and will provide details on the test methods and procedures used to determine the way in which the criteria have been satisfied for the specified level of protection. In addition, the manufacturer must show the way in which the test procedures map to the development specification, or to the Top-Level Specification for systems requiring verification.

The final evaluation cannot take place until the manufacturer has completed his internal acceptance testing and the system is available for field testing, so that the evaluation team will have complete access to the system for hands-on testing. There is no requirement that the evaluation occur as soon as the system is available. The manufacturer may choose to wait for some future release of the system before the final evaluation takes place.

The manufacturer will perform the actual detailed testing and where necessary, verification, to clearly demonstrate the protection capabilities of the system. To aid the evaluation team's analysis of the testing, the manufacturer should provide the complete test plan and any test data requested by the evaluation center.

The evaluation team will determine what further testing is necessary, if any, to assure that the system provides the security and integrity for the specified target level, using the manufacturer's qualification testing as a starting point. The result of the final evaluation will be to determine the "actual" level of protection and to place the system in the evaluated products list. The output from the final evaluation will be in three parts: 1) a public document giving the level of the system and the possible environments where it is usable; 2) a classified flaw

analysis of the system, including limitations and vulnerabilities, and where and how the system can be used; and 3) evaluation team notes.

3.4.3.4 Periodic Re-Evaluation

Computer systems, being dynamic, will be modified or enhanced at random intervals and thus will require re-evaluation. The evaluation center and manufacturer will jointly analyze all system changes to determine the security related aspects and thus the extent of the re-evaluation needed. The higher the level of the system, the more detailed the re-evaluation will be. For example, code related changes may only effect systems of level 5 or higher where code proofs are required, while design changes will effect systems of level 4 or higher since these systems require mathematical proof of correspondence of the design to a security model.

3.4.3.5 Timing of the Request for Evaluation

The evaluation of an industry developed computer system may start during any phase of product development. As part of the evaluation process, the center hopes that its insight and feedback to the manufacturer will tend to enhance the trustworthiness of the final system. Because of this, we believe that the earlier in the cycle the evaluation is started, the greater the protection potential for the resulting system, since the security design will be reflected in all specifications, and because there will be maximum exposure between the development team and the evaluation center. In conflict with the idea of early contact is the need for adequate system definition and the need to minimize exposure of the manufacturer's sensitive marketing plans. Ideally, the request will occur in the early stages of product development but after the system design has been well defined in terms of the TCB.

It is important to note that high-level design information which is usually produced in the early phases of development may not exist when evaluation is started later in the cycle. Since this information is essential to a proper evaluation, the manufacturer may find it necessary to produce specifications after-the-fact. This will only happen for systems designed for a high level of protection.

3.4.3.6 Configuration Management

The manufacturer must provide a physically secure facility where a master copy of the evaluated product will be maintained (for products of level 4 and above). This is needed to provide some assurance to the user of a trusted system that the copy he receives is a true copy of the

system that was evaluated. For some high level products, the manufacturer will be required to provide a secure machine facility for development and testing of the trusted system.

3.4.4 Summary

A process for evaluating the security and integrity of a commercially produced computer system has been presented. In a sense, the process is both generic, in that generalized evaluation factors have been defined, and specific in that seven protection levels are used to categorize the evaluated system. The process is defined in terms of a TCB, but is readily extendible so that future systems of possibly different configurations can also be evaluated. Implementation of this evaluation process will require the cooperation of both private manufacturers and the government-wide computer security evaluation center.

3.5 CURRENT EVALUATION EFFORTS

Prior to the establishment of an evaluation center and firm evaluation criteria, there are a number of computer systems undergoing informal evaluation. Since a government-wide computer security evaluation center has not yet been established, the evaluations are being conducted by evaluation teams made up of Consortium members and the Consortium's technical support.

The evaluation teams meet with developers of the systems on a periodic basis. These meetings have allowed the evaluation teams to learn about the specifics of a manufacturer's product and have provided a forum for discussion of the evaluation criteria and process in the context of the manufacturer's proposed plans. Since the meetings often result in company-proprietary information being made available to the team members, a Technical Information Agreement between the DoD and each corporation is signed. This agreement spells out the Government's obligations in keeping the proprietary information from public disclosure. A copy of the Technical Information Agreement appears in figure 3-8.

The following sections briefly describe the computer systems currently under evaluation.

3.5.1 DEC VAX/VMS

VMS is DEC's operating system for their VAX-11 series of computers. It has been designed to make full use of the hardware features of the VAX machine. VMS uses all four protection domains (user, supervisor, executive, and kernel), and takes full advantage of the virtual and I/O mechanisms.

VMS can be decomposed into three portions, corresponding to the three non-user VAX-11 hardware domains. That portion of VMS typically thought of as the core operating system is itself decomposed to run in both the kernel and executive domains. The command language interpreter runs in the supervisor domain, while user programs run in the user domain. The size of the code running in the kernel and executive domains is estimated to be 65-100K bytes. The size of the kernel domain itself, not counting device drivers, is about 30K bytes. The remainder of this discussion will focus on the VMS core operating system in the executive and kernel domains.

The VMS kernel implements the abstraction of processes, and generally runs in the context of the process that caused it to run. Certain processes maintained by the kernel are operating system processes that run in domains other than

TECHNICAL INFORMATION AGREEMENT

This Agreement is between
(hereinafter the "Corporation") and the Department of Defense (hereinafter the "DOD").

The Corporation may disclose to the DOD certain documents containing technical information and ideas (hereinafter collectively called "Data") on present and potential future computer architectures and operating system designs that are believed by the Corporation to be confidential and therefore exempt from disclosure under 5 USC Section 552 (b) (4) (1976) (The Freedom of Information Act). The documents containing such Data are identified as

. The Purpose of this disclosure is to enable the DOD to evaluate the security and integrity of the computer architecture and design, and except as specified below the DOD will not disclose or use such Data other than for the purpose of evaluation as stated herein.

All Data which the Corporation believes to be exempt from public disclosure shall be marked with a proprietary notice as provided by the Corporation. At the time the Data is submitted to the DOD and on a quarterly basis thereafter the Data will be reviewed by the DOD and the Corporation to identify the Data or portions thereof that the DOD and the Corporation believe to be exempt from public disclosure under 5 USC Section 552 (b) (4) (1976) (The Freedom of Information Act). If at the time of submission of the Data to the DOD or as a result of a quarterly review, the DOD and the Corporation do not agree that any portion of the submitted Data is exempt from public disclosure, such portion of the Data and any copies thereof shall be immediately returned to the Corporation. The Corporation and the DOD recognize that Data which is agreed to be exempt from disclosure at the time of submission to the DOD or at the time of a quarterly review may subsequently cease to be exempt. Therefore, the Corporation agrees to notify the DOD when Data is no longer believed by the Corporation to be exempt from public disclosure. Failure to provide this notice, if in fact notice should be given, does not mean that the Data is still believed by the Corporation to be exempt.

The DOD will expend its best effort during the term of this Agreement to protect the Data or portion thereof that the DOD and the Corporation agree, at the time of the latest review, to be exempt from public disclosure under 5 USC Section 552 (b) (4) (1976) (The Freedom of Information Act), PROVIDED HOWEVER, that the DOD shall not be liable for any unauthorized disclosure or use if such Data:

(a) is presently known or hereafter becomes known to the public by other than a breach of duty hereunder, or

(b) is known to the DOD prior to the time of disclosure to it by the Corporation, or

(c) is subsequently developed by the DOD independently without reference to the Data, or

(d) is independently and rightfully acquired by the DOD from another source without restriction, or

(e) is identified by the Corporation as believed by the Corporation no longer to be exempt from public disclosure.

FIGURE 3-8

In the event that a third party requests Data covered by this Agreement, the DOD agrees to notify the Corporation and to consult with the Corporation in advance concerning release of the Data to any such third party. Further, in the event of such request by a third party, the Corporation agrees to support the effort of the DOD to protect the Data of the Corporation by providing information necessary to justify an administrative denial of the Data to a third party, and the Corporation shall endeavor to intervene in any court proceeding initiated by a third party requestor of the Data. If the court denies the motion for intervention the Corporation agrees to provide an amicus curiae brief for the consideration of the court.

In the event a court orders the DOD to release the Data to a third party requestor, DOD shall provide the Data to the third party in conformance with the order of the court.

This Agreement constitutes the entire understanding between the parties hereto with respect to the subject matter of this Agreement.

This Agreement will commence on the date of signing by the parties hereto and terminate years, subject to renewal. Upon termination of this Agreement, should DOD be in possession of any Data considered to be proprietary by the Corporation, DOD shall, at the corporation's option, either destroy said Data or return it to the Corporation.

FIGURE 3-8 CONCLUDED

user and perform functions such as file management and process swapping.

Within the kernel itself, there is a logical fifth domain that is called the interrupt domain. This "domain" consists of that kernel code which runs at the interrupt level in response to some kind of interrupt, and consists mostly of I/O drivers and some code to support the InterProcess Communication (IPC) mechanism. VMS supports two types of IPC mechanisms: mailboxes and common event flags. Mailboxes, similar to UNIX pipes, are supported by a rather complex mechanism that includes quotas. Mailboxes are made to look just like I/O devices, and are hence processed at the interrupt level within the kernel. Common event flags are a simpler, semaphore-like mechanism.

Kernel functions that run outside the interrupt level include the AST (software interrupt) mechanism, the PAGER, the QIO mechanism, and the mechanism portion of the scheduler.

3.5.2 IBM System/38

The System/38 is the latest IBM offering in a series of business computing systems. The series began with the System/3 and evolved through the System/32 and System/34. Initial customer deliveries of the System/38 were made in June 1980.

System/38 consists of a high-level base machine (hereafter called HLM), and three IBM licensed software products: Control Program Facility (CPF), Report Program Generator (RPG III), and Interactive Database Utilities (IDU).

HLM is an object-oriented high-level machine architecture providing many of the supervisory and resource management functions found in typical operating systems, including functions specifically designed to support data base processing. HLM is itself internally layered into the physical hardware, a horizontal microcode layer, and a vertical microcode layer. HLM is the focal point of this description of System/38 and is treated in greater detail in the following subsections.

CPF extends the object-oriented HLM architecture to provide typical operating system abstractions and functions to the user, and to the RPG III and IDU subsystems. CPF constructs such user-level abstractions as files, programs, directories, and message queues out of the more primitive objects implemented by the HLM. CPF provides integrated support for interactive data base and work station applications and for batch processing.

RPG III is an enhanced version of the RPG II programming language found on the S/3, S/32, and S/34 computers.

IDU consists of a set of data base utility programs which build upon the file management and data handling functions of HLM and CPF to provide interactive data entry; source language entry; and query, retrieval, and update operations.

3.5.3 Tymshare GNOSIS

Gnosis is a capability-based operating system for the IBM S/370 architecture. As a capability-based system, Gnosis controls access via hierarchies of capabilities, not by access lists. Gnosis is made up of a kernel and its supplementary software. The Gnosis kernel is unswappable and runs in supervisor mode with real addresses. The kernel does not include a file system and does not perform scheduling. It does not include language processors, an editor or a loader. These make up Gnosis's supplementary system which, when combined with the kernel, comprise a more 'normal' operating system.

The Gnosis kernel functions as a simple reference monitor: for every action requested, the requester must hold the appropriate capability. No capability, no access. The kernel is small (approximately 50K bytes), written in assembly language and can be described more as a control program rather than a complete operating system. It is designed to meet commercial needs for an efficient and secure (guaranteed isolation of users) basis for transaction-oriented applications. As a fringe benefit, the relative simplicity of Gnosis (as is the case with VM) permits easy conversion of programs written for other IBM operating systems.

Tymshare targeted Gnosis to the S/370 architecture, which they know well and use in their commercial applications. With IBM's introduction of its low-priced 43xx family and its existing 30xx series on the high end, there is now a formidable range of S/370-compatible machines, which is to some extend duplicated by machines produced by the plug-compatible manufacturers (Amdahl, Itel, etc.). Whatever its perceived shortcomings in the security area (for example, it is only a two-state machine: privileged and problem) the architecture is extremely popular and in widespread use.

Gnosis and VM are both control programs, but with different design goals. VM remains the only operating system on which other operating systems are developed and tested. Gnosis itself is being developed under VM. Gnosis, however, should be considerably better adapted than VM for transaction-based applications, which do not require the generality of the

virtual machine approach.

To understand intended applications for Gnosis, consider a large database on a computer shared by mutually suspicious users. Both need to use the database (and other applications on the computer) but each wants to be guaranteed that the other cannot monitor accesses to the database or tamper with private applications. Gnosis provides building blocks (capabilities) and per-process virtual environments (domains) from which Tymshare claims policies which guarantee isolation can be implemented. Unlike the DoD sponsored kernelized operating systems under development, Gnosis is not formally specified or verified.

3.5.4 Navy SHARE/7

The SHARE/7 Timesharing System was developed by the Naval Fleet Combat Directive System Support Activity San Diego (FCDSSASD) in the 1972-74 time frame with multi-level security as a principal design goal. SHARE/7 runs on the AN/UYK-7 military computer built by UNIVAC (no commercial counterpart), and is principally used for the interactive development and testing of CMS-2 language tactical software for the AN/UYK-7 and AN/UYK-20 fleet computers. The system provides a comprehensive set of interactive tools specifically tailored to development and testing, including file utilities, editor, compilers, loaders, debuggers, simulators, program documentation aids, and configuration management controls. FCDSSASD is responsible for distributing the system to other Navy commands. Fifteen Navy sites use the system.

The SHARE/7 system is a multiprocessor system. FCDSSASD has two interconnected 3-CPU AN/UYK-7 computers with two I/O Controllers (IOCs), 15 memory modules of 240,000 32-bit main memory, and a disk storage subsystem. A single, modified NOVA/D116E minicomputer is used as a terminal multiplexor for both systems.

The AN/UYK-7 is a two-state machine: a privileged interrupt (or monitor) state, and task (or user) state. It supports both memory mapping and protection hardware: its descriptor-based process. Each segment is composed of 512-word (32-bit) pages and can grow as large as 128 pages or 64K words, making the maximum address space of a process 512K words. The protection hardware distinguishes between read/write, read-only, and execute-only access.

I/O appears to be interpretive, i.e., handled exclusively through system calls. There is no user or mapped DMA I/O.

The kernel of the operating system, called the Monitor, is responsible for handling all hardware and software (system

call) interrupts. When a user logs in, the Monitor creates for the user a single process (also referred to as "job" or "task") running a trusted, or privileged, executive (EXEC) command interpreter. EXEC then runs programs for the user in "sub-processes" within the original process. Each sub-process may recursively start other sub-processes, some of which may be trusted, thereby generating a stacking effect like procedure calls.

The Navy has initiated a Security Test and Evaluation (ST&E) to determine if the system will be accredited to operate in multi-level mode, running unclassified, confidential, and secret information simultaneously. The Consortium has been asked to provide support.

3.5.5 COINS II Terminal Access System

TAS provides a uniform processing environment for computer-naive intelligence analysts as they query COINS data bases distributed on hosts on the COINS II intelligence network. It is implemented as a COINS II network host on a PDP-11/70 processor operating under the UNIX operating system. The analyst logs on to TAS and uses TAS to formulate queries of data bases distributed on COINS II. He formulates these queries in the ADAPT query language. TAS takes each query and establishes a connection to the appropriate COINS II host; creates and submits a batch job (or, in the case of SOLIS, logs on and issues SOLIS commands) to perform the user's query; and receives the resulting response, apprising the TAS user of its arrival. All of this processing occurs transparently to the TAS user. TAS provides the analyst with a repertoire of commands to prepare, edit, and submit queries; to display and print responses; and to check on the status of pending queries. In addition, TAS gives the analyst facilities for creating and editing text files; for creating and sending mail to other analysts; and for retrieving and displaying mail.

TAS provides control of access to the COINS hosts. It maintains per analyst and per terminal data bases which establish analyst and terminal clearances and determine which COINS II hosts and data bases may be accessed by each analyst and terminal. TAS provides a subsystem (accessible only to the TASMASTER) for the maintenance of these data bases. TAS maintains a log of all network transactions and all access violations.

The COINS program office is investigating the impact of upgrading TAS and the COINS network to incorporate the DOD standard Transmission Control Protocol Version 4 with internetworking protocols (TCP4-IP) and converting TAS and NAS from UNIX to KSOS. The Consortium is supporting formulation of a multilevel secure design for TAS to act as

a basis for the impact analysis.

3.5.6 DTI COS-NFE

The Communications Operating System/Network Front End (COS/NFE) is a prototype network front end designed to connect a host (currently a WWMCCS H6000) and terminals to the AUTODIN II packet switching network. It is being developed by Digital Technology Incorporated (DTI) under contract to the Defense Communications Agency with System Development Corporation (SDC) as the subcontractor responsible for specification/verification tasks. Currently, the COS/NFE is targeted for implementation on a PDP 11/70.

Two major design goals for the COS/NFE are high message throughput and verifiable DoD multilevel security. The need for multilevel security arises from the fact that the terminals attached to the COS/NFE may be at different security levels and also from anticipation that a multilevel host may be used with the COS/NFE or attached to AUTODIN II.

The software architecture of the COS/NFE is not a fully functional operating system (no applications run on the COS/NFE), but is required to be a high speed, event driven secure protocol processor. The protocols that will be implemented in the COS/NFE are link level, channel level and service level Host-to-Frontend protocols (HFP), AUTODIN II THP, TCP, IDP, SIP and Mode VI protocols and various terminal protocols. Secure message processing is achieved by the identification and total separation of protocol processing for messages of different security classifications. In addition, the COS/NFE will maintain a database for authenticating users and their connection requests as well as producing a printed audit trail of all security related events.

To insure security, a complete formal specification verification of COS/NFE is planned. A top-level specification has been written and verified in SDC's Ina Jo specification language. At least two more lower level specifications will be produced and verified. Implementations of the COS/NFE will be done in the high level language PASCAL and code proofs may be done.

3.5.7 Intel 432

Discussions have just begun with Intel with respect to their 432 system. The 432 is an Ada-oriented machine that includes a significant amount of hardware support for high-order language and virtual machine concepts. The hardware and its "silicon operating system" treat processors, processes, storage, and interprocess messages as

objects. Resource management is supported by the silicon OS. The 432 is of interest because virtual environments such as those provided by the 432 are the basis for protection enforced by a TCB.

3.6 STATUS SUMMARY

This section of the report has described the prime elements of the Initiative and its evaluation effort. The concept of an Evaluated Products List was illustrated. We described the Trusted Computing Base: a set of generic requirements for the protection mechanism in a trusted system. The extent of Initiative involvement in workshops and seminars, including the Initiative-sponsored seminars directed toward industry, was detailed. Finally, we covered the proposed evaluation criteria and evaluation process, and we indicated what systems are currently undergoing an informal evaluation procedure to increase the government's and industry's understanding of trusted system development and evaluation.

The next section describes the status and plans for the technologies critical to trusted system development.

SECTION 4

RESEARCH AND DEVELOPMENT ACTIVITIES

There are three major efforts in the DOD computer security R&D program: (1) the development and demonstration of trusted general-purpose operating systems, (2) the development of applications of trusted computer systems, and (3) the establishment of a verification technology program to advance the state of the art in trusted system specification and verification. The sections below discuss the current R&D efforts and plans in each of these areas.

4.1 Trusted Operating Systems

The development of trusted operating systems was identified as critical to achieving overall trusted systems as far back as the Ware Report (see section 2.1). The trusted operating system area has now reached the demonstration stage. Early R&D efforts (1972-1976) produced the following results:

- A Verified Top-Level Specification (TLS) and implementation of a security kernel for the DEC PDP-11/45 minicomputer.
- A verified TLS for a security kernel for the Multics operating system.
- Two prototype security kernel implementations for supporting the UNIX operating system running on a PDP-11/45 or 11/70 minicomputer.

This section discusses current R&D efforts, ties current commercial plans into this work, and briefly indicates future directions for the trusted operating system area.

4.1.1 Current Research

Current trusted operating system R&D is focused on three systems:

- The Kernelized Secure Operating System (KSOS), a UNIX-compatible trusted operating system running on the DEC PDP-11/70 minicomputer.
- The Secure Communications Processor (SCOMP), and a version of KSOS that runs on the SCOMP (KSOS-6).
- A Kernelized version of the VM/370 operating system (KVM/370).

Each of these efforts is discussed below in some detail.

4.1.1.1 KSOS-11

The Kernelized Secure Operating System (KSOS) program began as a DARPA program, with funding from a variety of sources. Through the efforts of the Initiative, arrangements have been made, starting in October 1980, for the Navy to assume technical and contractual responsibility for the KSOS effort.

KSOS began as an effort to produce a commercial-quality security kernel for the UNIX operating system, running on the DEC PDP-11/70 minicomputer. The effort was named KSOS to reflect the fact that its kernel was independent of the UNIX operating system that is considered proprietary by Western Electric. This effort is often referred to as KSOS-11 to distinguish it from the KSOS being developed for the Honeywell SCOMP minicomputer (see section 4.1.1.2).

The KSOS system is being developed by Ford Aerospace and Communications Corporation (prime contractor), and SRI International (subcontractor in the verification area). Prototype secure UNIX designs done by UCLA [POPE79] and MITRE [WOOD79] were provided as inputs to the KSOS design, but the final design is different from both the MITRE and UCLA designs, although it is closer in architecture to the MITRE design.

Ford has prepared top-level specification of the KSOS kernel in SPECIAL that are being verified to an SRI restatement of the Bell and LaPadula security model. KSOS was originally to have been coded in EUCLID, but the lack of an operating EUCLID compiler within the necessary timeframe forced a switch to the Modula language. SRI has built some tools to do correspondence proofs between Modula and the SPECIAL specifications, but full code proofs for a system the size of KSOS are beyond the state-of-the-art for current verification technology.

The KSOS system is divided into three main portions: the security kernel, the UNIX emulator, and the Non-Kernel Security-Related Software (NKSR). The kernel runs in the PDP-11/70 kernel mode, and supports the following object types: processes, segments, files, devices, and subtypes. Files are organized in a flat file system by the kernel. UNIX-style directories of files are implemented by part of the NKSR called the Directory Manager. Subtypes are used as a type extension mechanism.

The Emulator uses the KSOS kernel primitives and maps them into the environment expected by a program running UNIX. The NKSR implements TCB User Functions (see section TCB.4.2

in section 3.2) through a secure terminal interface that allows the user to communicate reliably with the NKSR through the kernel. This type of interface is extremely useful in preventing certain types of spoofing.

4.1.1.2 KSOS-6 (SCOMP)

The Honeywell Secure Communications Processor (SCOMP) is a modified version of the Honeywell Level 6 minicomputer. The hardware modifications, made with security in mind, were first conceived to address the need of a secure front end terminal controller for the Multics system. In 1977 DARPA and various other organizations began funding the SCOMP hardware development and the development of a UNIX-compatible security kernel for the SCOMP hardware. Starting in October 1980, the Navy assumed technical and contractual responsibility for this effort. This version of KSOS is often called KSOS-6 to distinguish it from the PDP-11 version of KSOS (KSOS-11).

The SCOMP hardware consists of a standard Honeywell Level 6 minicomputer with a Security Protection Module (SPM) added. The SPM provides segmentation, paging, protection rings similar to Multics, argument validation, and virtual address translation for I/O programs.

In addition to developing the SPM, Honeywell is developing a version of KSOS that will run on the SCOMP hardware. KSOS-6 differs from KSOS-11 primarily where there are hardware differences between the 11/70 and the SCOMP. The SCOMP supports a large number of small segments, as opposed to the PDP-11's eight segments. However, the SCOMP segments are very small compared to Multics segments that are up to one megabyte each. KSOS-6 memory management is therefore quite different from KSOS-11. The SCOMP supports virtualized I/O devices that are not shared. As a result, the KSOS-6 kernel supports an additional object, namely devices, that the user can program directly.

KSOS-6 uses the protection rings of the SCOMP hardware. Calls from the user to supervisor and supervisor to kernel rings, as well as argument validation, are done in hardware.

KSOS-6 has been specified in SPECIAL and will be coded in UCLA Pascal. The top-level specification is being verified using SRI's Hierarchical Development Methodology (HDM).

4.1.1.3 KVM/370

IBM's Virtual Machine/370 (VM/370) operating system is designed to run on IBM medium-to-large-scale computers. A virtual machine structure provides separate "virtual environments" for any number of users. Each of these

environments provides a set of processor resources that is the same as those available to a user if he were running on the "bare" machine. Thus, a virtual machine multiplexes the physical processor resources and isolates and protects each virtual environment from the next.

In March 1976, DARPA sponsored an effort at the System Development Corporation (SDC) to produce a kernelized control program for VM/370. The resulting system is called KVM/370 [GOLD79]. This effort is scheduled to produce an initial system release to test sites in the first quarter of 1981. Milestones along the way have included a determination of the feasibility of the project through an abstract design in the first year, and an initial demonstration at the end of the third year. The KVM effort is now funded and managed by RADC.

KVM/370 offers each user the capability to share the physical processor among the same (or different) versions of an operating system, each running at different security levels. A limited amount of information sharing between virtual environments is also possible with KVM/370, constrained by the DoD rules for information access.

The KVM/370 system architecture consists of the following:

(1) A kernel and verified trusted processes, running in the real supervisor state of the S/370.

(2) Audited semi-trusted processes having access to same global system data, executing in real problem state, but having access only to virtual addresses.

(3) Non-Kernel Control Programs (NKCPs), one per security level, having access to system data for the supported security level only, executing in real problem state with virtual addresses.

(4) User VMs, each controlled by the appropriate NKCP for its security level, executing in real problem state.

Great pains were taken in the design of KVM/370 to provide correct mediation of I/O access by the KVM/370 kernel and trusted processes. The problems arise because the I/O Channel Control Words (CCWs) must be statically analyzed for security correctness before allowing the channel to operate on them. The hardware offers no help in this area.

The kernel and verified trusted processes were originally intended to be written in a strongly-typed PASCAL-based programming language such as EUCLID, but because of its lack

of availability at the time, JOVIAL/J3 was chosen instead. JOVIAL is not a verification-oriented programming language. Consequently, although the formal specifications of KVM/370 will be verified against the security policy enforcement criteria, the implementation will not be formally verified.

4.1.2 Status of Industry Efforts

Past R&D in the area of trusted operating systems has focused on security kernel and trusted processes as a means of achieving a high level of trust in the security features of an operating system. The efforts described in the previous subsection have paved the way for the computer industry to develop their own trusted operating systems. But the computer industry must deal with one salient problem that the R&D community has been able to avoid: installed customer base. A computer manufacturer's installed customer base is both his source of current income and the base for any further market penetration. It is the largest single determinant of future share of the market. This is the case because customers build up a tremendous investment in personnel training and application software as their personnel use a particular manufacturer's computer system. With each upgrade in system, where there is the possibility of moving to a different manufacturer's computer line to take advantage of new features not present in the product line of the current supplier, this investment argues strongly to stay with the current supplier's product line. Thus, a manufacturer is reluctant to make changes to his product line that will lead his customers to question his commitment to continued support of the base for their applications.

The inertia represented by the installed customer base works against rapid and easy assimilation of the TCB-based protection technology into available computer systems. The computer systems that dominate the market today have their design rooted in the era of batch job processing. Protection in modern processing environments has to be based on the reality of timesharing: each user of the computer system is, in effect, a separate job stream. Most large-scale systems have adopted this model of interaction at the user level, and changes have been made in the underlying hardware to incorporate changes in technology, but a substantial portion of the system is carried over from the old architecture.

The TCB protection concept argues that protection be based on a virtual environment per user and that the mechanism enforcing this protection be central to the design. The bare computer is multiplexed (timeshared) into a separate virtual computer for each user. The user-level features of the bare computer (sometimes slightly enhanced) are made

available to each user's operating system. Details of the tables used to maintain and enforce these virtual environments are hidden from the user. The operating system is essentially a single-user operating system.

Operating systems that derived from the batch era do not satisfy these requirements. Most started out as single-user (or rather single job stream) operating systems. In many cases, if more than one user is to be served, the multiplexing among users is done in the application software for that application. Where a "user virtual environment" has been provided, it is usually provided in many different parts of the large operating system and, therefore, typically scattered through more than a million lines of code. In these systems, a great deal of the internal structure of the operating system (e.g., tables) is visible to the user or application software. Application software takes advantage of the availability of these internal data structures, often for performance reasons. Thus, in existing computer systems, we see inter-user protection dependent on large portions of the operating system and application software dependent on numerous internal operating system data structures.

In recognition that timesharing suggests a different computational model, and for other reasons of reliability alluded to in section 1, manufacturers' computer architects are gradually working toward the virtual environment per user model. However, demands for compatibility with existing application software generally outweigh the architects' desires for a more simple (and therefore more maintainable) system. Furthermore, in addition to the customer dependencies described above, we see resistance to change on the part of the system software people inside a computer company. They have their own investment in familiarity with the internals of the existing operating system. Both Honeywell and IBM have systems that incorporate a reasonable timesharing model in the form of Multics and VM/370, but these systems constitute only a small part of their respective sales--even though they have been available for ten years.

The industry relations portion of the Computer Security Initiative must take into consideration the manufacturers' concerns. It is very unlikely that a manufacturer will make significant changes to his new product lines unless he can somehow maintain compatibility with the existing application software. And it is likewise unlikely that he will make changes toward better protection in his systems unless there is a strong incentive from the marketplace.

The concept of common evaluation criteria as embodied in the TCB proposals currently undergoing industry coordination and

the idea of a government-wide evaluation center aim at generating market pressure.

Although it should be clear from the foregoing that current operating systems employ an architecture based on a model of processing and protection that is quite different from the timesharing model desired today, one would still like to be able to evolve from current systems to ones that provide better protection and integrity. Based on past and ongoing experiences, we see two ways for such an evolution to occur: radically new systems can be introduced that satisfy protection needs but can present an environment that will allow the customer to continue to run his favorite operating system or applications; or the user interface can be evolved toward a consistent protection model and the interface to the operating system gradually solidified and made opaque with respect to internal data structures.

The totally new system approach has already been taken for Multics and VM/370. Both systems are capable of acting in modes that allow existing mainline operating systems of the respective vendors to run as if they were on the native machine. Multics runs the GCOS operating system "encapsulated" and even runs large GCOS jobs faster than GCOS runs them on its native machine because of more advanced memory management hardware and software. VM/370 provides multiple virtual IBM S/-370s, any one of which can look to IBM's various S/370 operating systems like a bare S/370. In fact, VM allows a bare S/370 of one hardware configuration to look like another configuration. Further, the underlying architecture of the latest member of the IBM System/3x product line varies radically from the that of its predecessors, yet it supports the same customer community.

The evolution method is being pursued by at least two manufacturers. One of these manufacturers has added protection at the file and user level that would tend to support the DOD protection policy in a benign user environment. The other manufacturer has come up with a consistent set of access control mechanisms for protecting all objects accessible at the user interface, and has proposed to implement the mechanisms. The initial mechanisms would allow access to operating system data structures to be controlled as well as access to other users' objects. Eventually, when control of access to operating system data has been sufficiently restricted, the operating system can be restructured into a TCB and non-protection-relevant operating system modules. A third manufacturer has recently released a new product which already has a strong protection mechanism in place.

While none of these five systems was constructed using the verification technologies discussed in section 4.3 below,

each system provides better and more reliable protection than its predecessors and, in some cases, than other general-purpose systems. The increased protection stems from existence of a consistent protection model, improved simplicity and centralness of the protection-relevant structure of the system, and application of better design methodologies.

4.1.3 Future Directions

The three government developed kernel-based prototypes provide an adequate demonstration base for the feasibility of TCB-based trusted systems. These efforts will be extended for use in a wide variety of applications and to further explore the basic principles of security kernel development. However, the basis for most future work in trusted operating systems will come from individual manufacturers in the context of their own product lines and product plans.

Through the different aspects of the Computer Security Initiative, members of the manufacturers' design teams are gaining an appreciation of government protection needs and research in providing the needed protection. With this appreciation has come a receptive environment for inclusion of protection features in the list of customer requirements from which future releases of current products and future products are defined. Likewise, protection requirements and evaluation criteria from the Initiative effort are finding their way into RFPs for systems requiring protection of the imbedded computer system, and vendors are responding.

State-of-the-art development and verification methodologies are not yet a part of manufacturer offerings, although several of the manufacturers who are committed to real protection assurances (e.g., level 4 in the evaluation criteria) are actively pursuing verification possibilities. As verification technology moves into the production world, manufacturers will be prepared to adopt it as well if the requirement is firmly stated in the marketplace.

4.2 TRUSTED APPLICATIONS

The understanding and state of construction of production-quality prototype trusted computers incorporating the TCB concept has progressed to the point where feasibility is no longer a question. As indicated previously, industry has started to get involved in the construction of trusted systems, and the state of verification technology suggests that these systems must be structured in terms of a TCB and untrusted operating system software. It is now important to refine our understanding of areas where these systems can be applied and how the structure of the application software is influenced by the user virtual environment enforced by the TCB. Past R&D in trusted operating systems have provided two major benefits in terms of trusted applications:

- the ability to run trusted application code on an operating system that cannot itself be easily subverted (running trusted code on a conventional operating system can give a false sense of security because, even if the application code is verified, it can be subverted by attacking the operating system itself); and

- the verification technology developed for operating systems applies very well to small applications that do not require a general-purpose operating system.

This section reviews current trusted application efforts and discusses their implications for industry.

4.2.1 Current Research

This subsection discusses the major classes of trusted applications currently being studied and reviews current R&D efforts in these areas. The three classes are:

- Trusted connection of existing untrusted systems (guards)

- network front ends

- database management systems

4.2.1.1 Trusted Connection of Existing Untrusted Systems

Because of the present lack of general-purpose trusted computer systems, data of different security levels must be processed on different computers. Evolving defense systems depend heavily on the capability of passing information between computers operating at different levels. Unfortunately, such connections are very difficult to

implement in a trusted manner. This problem will be made much easier when all computers have trusted operating systems, but such is not currently the case.

There is a recurring need to make a subset of classified data in a given system available for use at a lower classification level, a process referred to as "sanitization" or "downgrading."

Currently, there are several methods available to provide a sanitize and release function. The simplest involves reading information to be sanitized from a terminal or the higher level sensitive system, and entering the sanitized version on a terminal or the lower level sensitive system; there is a manual "air-gap" between the systems. A more sophisticated solution involves a single CRT terminal that can be connected to either system (by means of a hardware switch), but to only one system at a time. In this mode, data from the higher sensitive computer is read into the terminal and sanitized using the terminal's editing capabilities. Then the terminal is disconnected from the higher sensitive computer and connected to the lower level system, and the data is read into that system..

The problem with these solutions is that they are time-consuming and cumbersome. A better approach involves providing a trusted interface between the two computers at different sensitivity levels. We call a trusted computer that implements this interface a "Guard." The Guard system allows data to flow between these systems in a secure and controlled manner. The Guard systems under development today have at least one human reviewer involved in the sanitization process. A Guard system can provide better throughput than the switched terminal described above, and is much more flexible in its capabilities. Two types of Guard applications have been proposed: (1) those that run trusted application code on a trusted operating system, and (2) those that run stand-alone, without an operating system. The following sections describe Guard systems currently being designed or developed.

4.2.1.1.1 The ACCAT Guard

The ACCAT Guard is a system that is installed at the Advanced Command and Control Architectural Testbed (ACCAT) at NOSC in San Diego. The ACCAT Guard has been built and tested on the UNIX operating system, and will run on KSOS in the near future. It has been built by Logicon Inc, under contract to the Navy.

The system provides an interface between two computers or networks operating at different classification levels.

Hereafter, the computer or network operating at the lower classification will be referred to as the LOW computer or network, and the computer or network operating at the higher classification will be referred to as HIGH.

The Guard allows these two computer/networks of different classifications to communicate by providing (1) an upgrading facility to pass data from LOW to HIGH, and (2) a sanitization and downgrading facility to pass properly sanitized data from HIGH to LOW with appropriate human review.

Two general classes of data transfers are provided by the ACCAT Guard. The first is ARPANET network mail transfers. Network mail can normally flow among users on the LOW network or among users on the HIGH network, but cannot flow between the two networks. ACCAT Guard allows mail to pass between the LOW and HIGH networks in a secure, controlled manner.

Second, the Guard allows users on the LOW computer/network to query a database resident on some HIGH computer, and allows properly sanitized responses to be sent to the requesting user.

The Guard minicomputer is connected to the LOW and HIGH networks through Private Line Interfaces (PLIs) [BBN77] over the ARPANET. PLIs are encryption devices that allow a computer with a specific key to securely communicate with other computers having the same key. Thus the Guard system has two distinct ARPANET connections that are at different security classifications. Figure 4-1 shows the connection of the Guard computer via PLIs and the ARPANET to the other computers. The LOW and HIGH computers are prohibited from directly communicating because their keys are different.

There are two types of personnel designated to operate the ACCAT Guard system, sanitization personnel and the Security Watch Officer. The main responsibility of the sanitization personnel is to sanitize responses from the HIGH database. A Guard system can have many sanitization personnel, whereas it has only one Security Watch Officer, whose function is to review all data downgraded by the Guard and approve or deny the downgrade. Thus the Security Watch Officer has responsibility for the security of the system.

The sanitization function can be performed by untrusted code operating at the high level and thus ease the verification burden. Therefore, all data is shown to the Security Watch Officer before it is released. The Security Watch Officer communicates through his terminal to a trusted process on the Guard that securely shows the data to be released and then asks for an acceptance or reject of the downgrade.

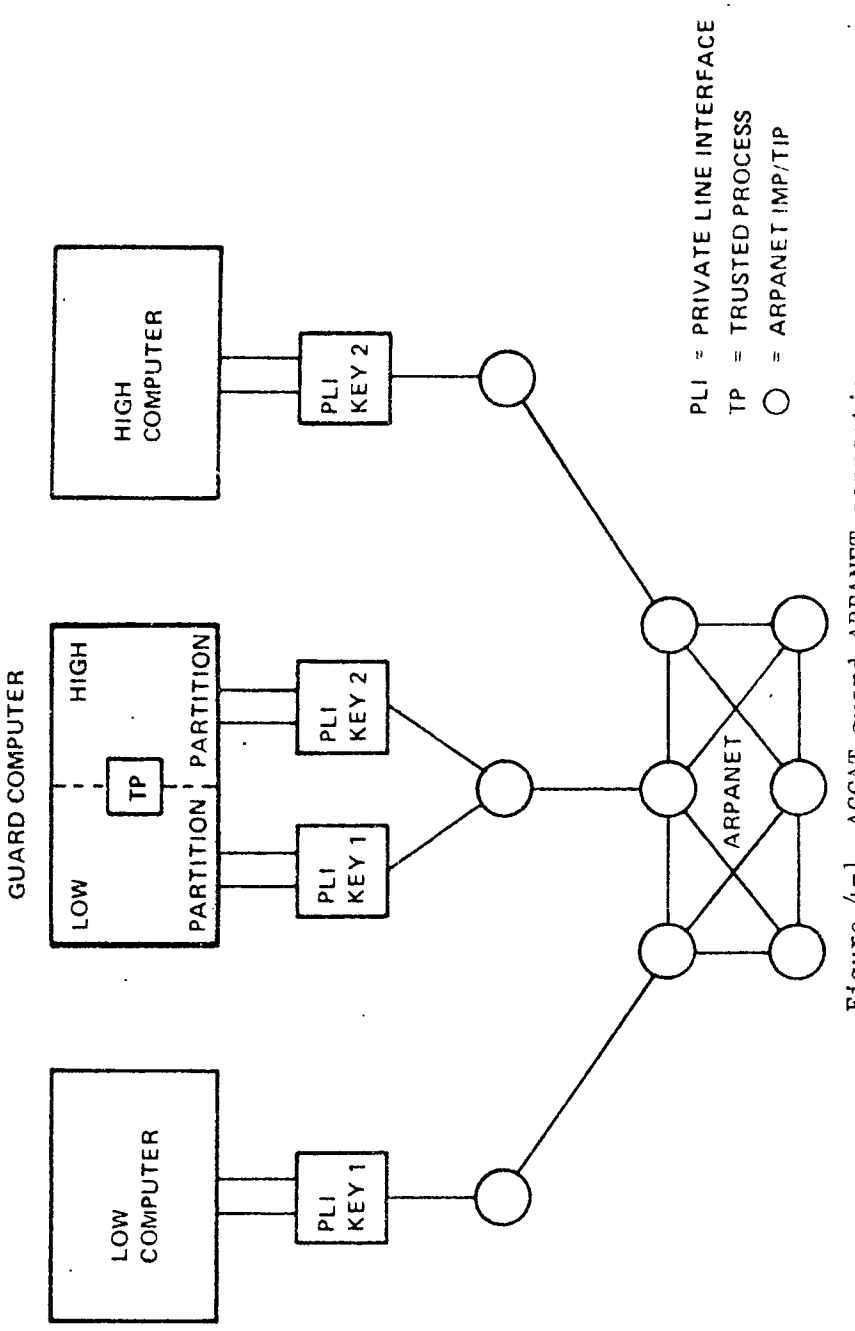

Figure 4-1--ACCAT guard ARPANET connections.

This trusted process is trusted to downgrade only data it has shown to the Security Watch Officer, and only at the explicit request of the Security Watch Officer.

4.2.1.1.2 LSI Guard

The LSI Guard system is a microprocessor-based Guard system, with functionality somewhat less than the ACCAT Guard. The LSI Guard is also a Navy program. The LSI Guard differs from the ACCAT Guard in that:

- it runs stand alone on a microprocessor without an operating system;
- it supports only one type of data transfer, similar to the mail transfer of the ACCAT Guard;
- it supports only one user, whose job is similar in function to the Security Watch Officer, but with an additional editing capability;
- it uses a sophisticated function key box to make the user's job easier and less error-prone, as well as easing the verification task; and
- all of the code in the system is expected to be verified.

A prototype function key box and implementation on a DEC PDP-11/03 (commonly called the LSI-11) have been built by the MITRE corporation. I. P. Sharp Associates Ltd. is building a verification environment combining the Stanford Verifier and the Euclid language with which they expect to develop a verified version of the LSI Guard.

4.2.1.1.3 The FORSCOM Security Monitor

The U.S. Forces Command (FORSCOM) is an Army operations center that is soon to become a node on the WWMCCS Intercomputer Network (WIN). A consequence of this upgrade is that FORSCOM must raise the classification of its operations, both at FORSCOM and its remote sites, from Secret to Top Secret, even though little or no actual Top Secret access is required.

The FORSCOM Security Monitor program seeks a near-term method for allowing Secret users of the WWMCCS Entry System (WES) to gain controlled access to the Top Secret FORSCOM WWMCCS/WIN system. The proposed approach is to identify a useful subset of the WES capabilities that are judged "safe" and place a guard-like system between this safe subset and the user. The current plan is to have the FORSCOM Security Monitor (FSM) ready to test in an operational exercise

scheduled for the fall of 1980.

The FSM system has been implemented on standard UNIX by Logicon Inc., with Navy and DCA sponsorship. The MITRE Corporation is preparing a design for the system that will minimize the amount of code that has to be trusted in providing the required functions.

The FSM system is similar in concept to the previous two Guards, though it works between users and a computer system rather than between two computers. Another unique feature of the FSM is that it "filters" the users commands so that only the "safe" subset of WES commands can be executed. Also, in addition to traditional review and release capability, the FSM can also automatically downgrade fixed-format responses to the user, such as error messages generated as the result of an erroneous command entry.

FSM-like applications have shown up in other systems. NASA is considering the use of an FSM-like approach for protecting Space Shuttle planning data about DoD missions.

4.2.1.2 Network Front Ends

Another class of trusted applications addresses some of the problems involved with trusted networking of computers of different classifications. Two such applications, the Terminal Access System and the COS-NFE, have been described in sections 3.5.5 and 3.5.6, respectively.

4.2.1.3 Database Management Systems

Using a trusted computer system allows one to process multiple levels of data without having to overclassify the data. If multiple levels of information must be intermixed and shared by the users of the system, the data must be organized and accessed in a meaningful manner, using a Database Management System (DBMS).

For a DBMS to securely process multiple levels of information, we must be able to trust the DBMS to protect the data. Therefore, multilevel database management is an important application for trusted computing systems. Two major trusted DBMS studies have been undertaken in the past under Air Force sponsorship, by I. P. Sharp Associates Ltd. [KIRK77] and System Development Corporation [HINK75]. Both of these studies helped to identify and clarify the key technical issues in trusted DBMS technology.

Two major conclusions and areas for future work can be drawn from this past work:

1. Using conventional, general-purpose TCBs to support a Trusted DBMS, protection of very small objects (e.g. individual fields in data records), although highly desirable, is difficult, often forcing designs that cause groups of records to be classified at a single level.

2. Even if multilevel records are supported, using them is difficult with the rules of the traditional mathematical security models.

The MITRE Corporation, under Navy sponsorship, is beginning a Trusted DBMS program that examines these conclusions. The MITRE approach involves the definition of a new (or modified) security model for trusted database management that is specific to a class of DoD applications. Rather than using a general-purpose TCB, MITRE will design a TCB designed to support an existing DBMS in a trusted manner. The design will be targeted for implementation on the SCOMP minicomputer (see section 4.1.1.2).

4.2.2 Future Directions

As just discussed, there are three areas of active application development. The first class contains guard systems that protect a large, untrusted computer or collection of computers from unlimited access by an uncleared or untrusted person. These systems are seen as a necessity in the near term to compensate for the lack of general purpose trusted computers. Certain guard systems may always be desirable, even with the widespread availability of trusted computers. Guard systems are being built on TCB systems or are restricted enough in function that the whole guard system is to be verified. Several types of guard system are under active development.

The second class of systems consists of communication processors and user support systems. These systems are complex enough that they need to be structured with an identifiable TCB. While they are intended to perform true multi-level functions, they may be used in the near term to serve guard functions as well. There are three advanced development applications and two production applications in this class that have been or are being directly influenced by trusted computer technology. COINS terminal access system (TAS), the Communications Operating System NFE (COS/NFE), and a Communications Access System (also for COINS) comprise the advanced development systems in this class. The Production systems are Autodin II and the SAC Digital Information Network (SACDIN). Two of the advanced developments have been or will be structured to run on KSOS. The other advanced development system and the two production systems have their own special-purpose TCBs.

The third class is Trusted Data Base Management Systems. A TDBMS can be seen as either a special-purpose operating system or an application. The TCB concept has a small effect on overall system performance in a general-purpose system because the TCB need only enforce data protection to the file and process level. All information within a file is protected at the level of the file, and each user process is restricted to modifying data at one classification. These restrictions are reasonable for most general-purpose computers because one is generally concerned about providing multiple single-level working environments for the users. These restrictions cause applications to be structured differently where that user is moving among protection levels (as in a message system), but the performance is still within reasonable bounds. However, in a data management system, every query will generally access data of at least two protection or sensitivity levels. Here, the conventional file-level protection would inflict a heavy penalty on either performance or ability to adequately discriminate between protection levels. For DBMSs, support for smaller data storage entities is needed in the TCB.

A program has been defined and is underway to explore the operational requirements for TDBMSs, examine the impact of these requirements on the current protection model, and design a TDBMS with a TCB tailored to DBMS protection needs.

With the widespread availability of verified, commercial trusted computer systems capable of supporting multiple DOD classification levels (protection level 4) still a few years away, interim solutions are needed. For those organizations that cannot structure their system so that a guard function can be used, there is an alternative: the computer on which the system will be implemented can be chosen from a product line where the manufacturer has made a commitment to evolve toward a trusted computer system (as described in the previous subsection). Precedent for this is found in the ACCAT Guard and COINS TAS systems, where the initial system is built for Western Electric's UNIX operating system. ACCAT Guard was developed on UNIX for use on KSOS; TAS was developed and installed on UNIX and is being considered for redeployment on KSOS.

This approach has several advantages. Development of the application software may begin on the existing system, with a clear idea from the manufacturer of what he plans to support on the trusted system. The existing system serves as both an in-place testbed and an indication of the compatibility base of the manufacturer. At the same time, the existence of a real application development targeted for a manufacturer's proposed trusted system will help improve the manufacturer's commitment to the system. He will see a firm market for trusted systems and will be encouraged by

the marketing value of being one of the first. The interaction between the application developers and the manufacturer will tend to further increase understanding of the issues and increase the amount of creative talent involved in the solution.

4.3 SPECIFICATION AND VERIFICATION TECHNOLOGY

Previous sections have referred to the role of specification and verification technology in establishing the assurance of the integrity of trusted systems. In this section, the technology is described and a plan for further development and transfer of the technology into a production environment is described.

4.3.1 System Components

Specification and verification systems are built around a language--a formal notation--ad typically provide at least the following components:

- specification processor;
- Verification condition generator; and
- theorem prover.

Specification languages, in this context, are formal notations related to mathematics, logic, and programming languages. Their purpose is to state precisely, and in only as much detail as relevant, the functions provided by a system of programs to be verified. For the sake of compatibility with software tools. the syntax of these languages is adapted to machine processing.

The scope of specifications, within the spectrum from abstract system properties to assertions embedded in programs, varies considerably among the systems surveyed. There is some tendency to use the word "specification" to include everything that is not in the implementation language, but we shall use "specifications" in a narrower sense to mean "functional specifications" unless otherwise indicated.

A specification processor may do nothing more than syntax checking, or it may also act on the basis of security requirements to generate theorems. The various ways in which security requirements may be stated as specification properties and translated into theorems are discussed below.

A verification condition generator (VCG) takes a program as input, together with some assertions about it, and uses knowledge of the programming language semantics to generate formulas. Assertions are not themselves theorems because they are not valid out of context; they become true only in conjunction with statements about the result of program execution, and these latter statements are generated by the VCG.

Specifications are the prime source of assertions, since they express design requirements. In addition, the user

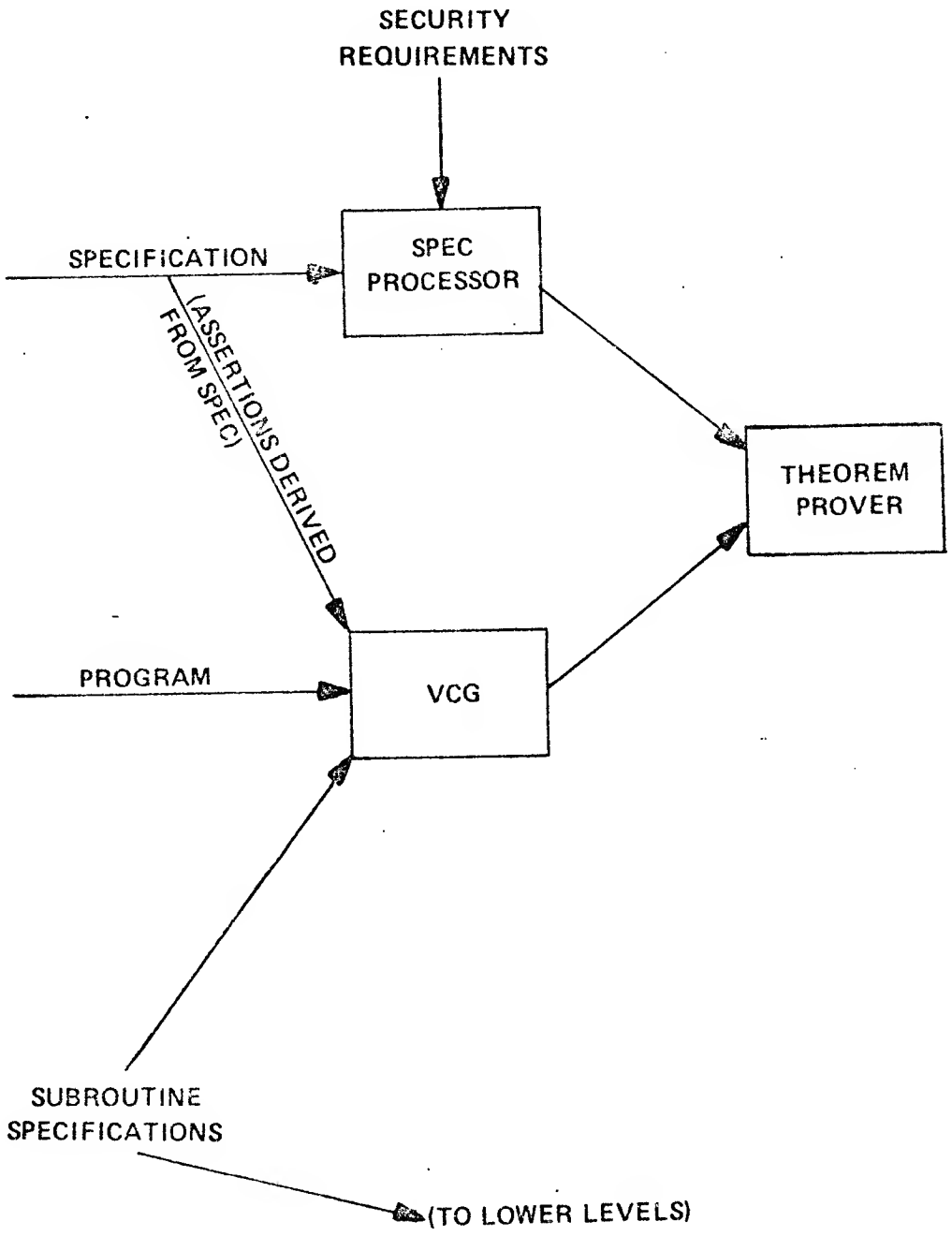

FIGURE 4-2. VERIFICATION SYSTEM COMPONENTS

must supply two other types of assertions.

Some additional assertions have to be supplied for technical reasons. An example is a "loop invariant" for each looping statement in a program. These are required in the verification approach used by all the systems investigated, but cannot usually be constructed automatically.

Of greater significance are the additional assertions required for subroutine or function calls. These assertions constitute a specification for the subroutines. They are the basis of a hierarchical approach to program development, since by giving the subroutine specifications, the calling program can be verified independently of the coding and verification of the subroutines. The collection of subroutines and their specifications ar the next lower level in the hierarchy, and they can be verified just as the calling programs are. The main difference is that security properties need to be checked only for the top level, or, in some cases, the second level.

Theorems arising either from a specification processor or a VCG are handed to a theorem prover. If a theorem prover is unable to prove a theorem with information at hand, it returns to the user, and an iterative or interactive process ensues, in which the user provides more facts, suggests a proof strategy, or goes back to fix either the assertions or the program.

Figure 4-2 summarizes the relationships among specification processor, VCG, and theorem prover, and shows where assertions enter into the verification process.

4.3.2 Security Properties

There are currently two types of security properties that have received significant attention in the context of verification of top level specifications: access control properties and information flow properties.

Access control properties are statements about the legality of a data structure representing a secure state, or about the transitions allowed between one state and the next. The *-property (restricting access to prevent copying information into a lower level object) is a state property. Tranquility (the security level of an active object does not change) is a transition property.

State properties are invariants; they are proved inductively by showing that if they are true for a given data structure, then they are true for the transformed data structure resulting from a function call. The specification processor

must be able to set up the induction step of the proof.

Transition properties do not require an induction step, but, like state properties, require a processor that can eliminate the time element from a specification to produce logical (timeless) theorems. Both of these types of properties relate to specifications in much the same way that assertions relate to programs.

There is really only one information flow property: information flow from one variable to another should be consistent with the security levels of the variables. A flow processor determines potential information flows from the specification by a syntactic analysis of the effects of function calls. The user contributes only the security level assignment. Thus, a flow processor is a specialized tool quite different from processors for transition properties or state invariants. Among the systems investigated, only HDM has a flow processor, referred to in that system as the "Multilevel Security Formula Generator".

4.3.3 Current Research

The MITRE Corporation prepared a survey of specification and verification methodologies for the Initiative. This subsection contains the introduction to that survey, and an overview of each of the systems surveyed. For more detail on the individual methodologies, see [CHEH80].

This survey was prepared at the request of the Chairman of the Computer Security Technical Consortium to survey and evaluate the status of prominent automated specification and verification methodologies. We have chosen, largely on the basis of availability of information, to discuss the following five systems in this report:

Gypsy	University of Texas at Austin
HDM	SRI International
FDM	System Development Corporation
AFFIRM	University of Southern California, Information Sciences Institute
Stanford Verifier	Stanford University

The methodologies discussed in this report should be considered experimental. For the most part, they are undergoing continuous evolution. Some groups have made the tools and documentation of their methodologies publicly available. However, none of these methodologies should be considered final products. Significant work, both theoretical and practical, remains to be done in most areas of software verification, and the systems can be expected to change as new research yields better solutions to the problems. Because this report is intended to assist in the

practical application of these methodologies, work in progress and future work is generally not discussed except in those cases where near-term solutions are evident, for example, a tool almost completed. Our assessments generally reflect the state of the systems as of late 1979.

In addition to aiding the verification of a system, application of one of these methodologies to a software development program may provide benefits in areas such as improved documentation, reliability, and maintenance. Indeed, some of the systems focus more on the overall software development process than on verification. Though we recognize these additional benefits may be as important as the verification itself, the scope of this report is limited to the specification and verification aspects. Particularly, we concentrate on proof of security of a design, correspondence of the design and implementation, and the degree of automation provided within each system. Experience and configuration data are provided for each methodology.

It was our intent that this report provide information encouraging others to explore the application of the methodologies we describe. For that reason, we concentrate on describing each of the rather different techniques independently. We have concentrated on surveying, rather than evaluating the five methodologies. We believe that considerably more experience and improvement is needed in the technology before a comparative evaluation can be meaningful.

4.3.3.1 The Hierarchical Development Methodology (HDM)

HDM is SRI International's approach to software development that covers the areas of system design, specification, implementation and verification, and provides a set of languages and tools to support these areas. The goal of HDM is to structure the software development process in a way that will result in highly reliable, verifiable, and maintainable software. Although the ultimate goal is to produce software that is completely verified, the verification tools of HDM are less complete at this time. Considerable work has been done in the area of verification: the theory is fairly well understood and some support tools are available, but much work remains to be done in the area of code proofs. Verification is currently one of SRI's major effort in HDM. The information in this section is largely a result of discussions with SRI. Except for topics related to verification, most of the information in this section can be found in the HDM Handbook [LEVI79].

The process of system development is outlined in figure 4-3. Each step is named in a box, with lines connecting related

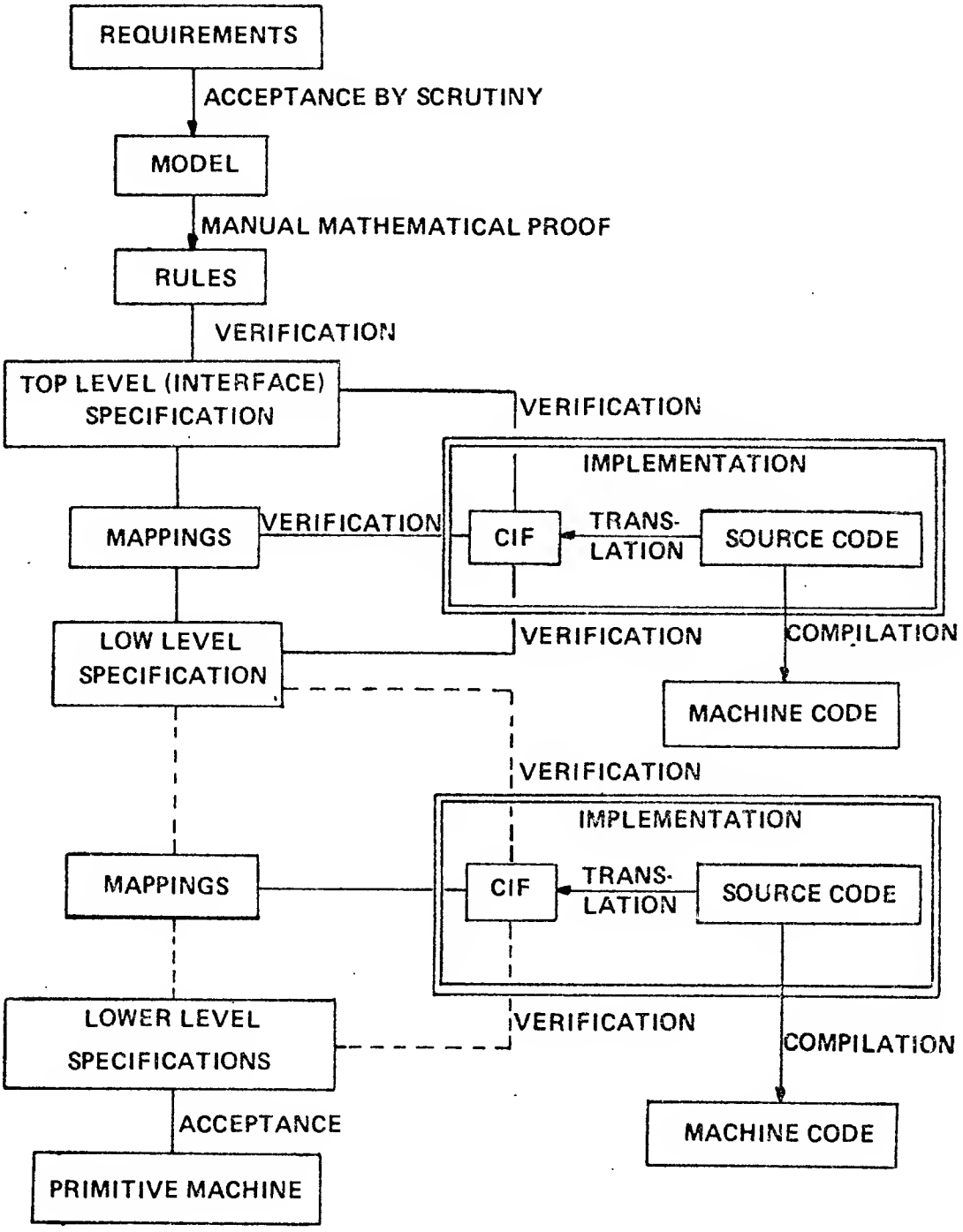

FIGURE 4-3. HDM SYSTEM DEVELOPMENT

steps labelled to show how the correctness of correspondence between steps is verified or proved. Arrows indicate where one step is derived from another. The entire development process is divided into two major stages: first is design definition and specification, shown running vertically down the left side of the figure, followed by the implementation stage on the right side of the figure.

The system definition and specification stage begins with the top two steps shown in the figure: preparation of a model of system requirements; and embodiment of the model as a set of rules of information flow. These two steps have, to date, only been carried out in an automated way for proof of security properties, although manual proofs of other properties have been carried out (see the discussion under DESIGN VERIFICATION below). Thus the requirements are the well-known subset of the Department of Defense security requirements that have been formalized in the Bell and LaPadula (MITRE) model of security [BELL74]. The rules of information flow are a restatement of the model in a more restricted sense so that proof of correspondence of the next step, the top level specification, to the model can be carried out automatically by a fairly simple technique called flow analysis. The top level specification, written in SPECIAL, should be the simplest possible description of the system's external behavior. The lower level specifications describe, at varying degrees of detail, abstract machines implementing more primitive functions of software and hardware, down to the lowest level that is assumed to describe the primitive machine--some arbitrary combination of hardware and software upon which the "verified" software runs. This primitive machine could consist of the machine language instruction set if the system were verified to that level, or it could consist of a set of higher level language primitives if verification stopped at the source code level. Between levels of specifications, mappings are written, in SPECIAL, that define how the data structures at a given level of specification are implemented in terms of data structures at the next lower level of specification. Note that at the end of the specification stage the only proof is that the top level specification obeys the rules of the model. The lower level specifications are simply design detail. However, all levels of specification and the mappings are subject to syntactic and consistency checks.

The next major stage of system development is implementation. This means writing the source code in some high order language for which there is a compiler that can generate machine code for the target machine. The choice of implementation language is unrestricted; however, if verification of the programs is desired, they must be translated manually or automatically into a Common Internal

Form (CIF), which is a simple programming language that can interface to the HDM tools. There is currently a CIF translator for Modula.

The implementation is structured as a set of procedures that show how each of the functions specified in a given abstract machine are implemented in terms of the functions in the specification of the abstract machine at the next lower level. Using the mappings and the two levels of specifications, the CIF representation of these programs is verified to be correct--that is, assuming the lower level machine works as specified, the higher level machine is implemented correctly. Given the correspondence of the CIF to the real programs, correctness of the CIF implies correctness of the real implementation.

SRI originally designed an "intermediate-level programming language" (ILPL) as an "abstract" language for the implementation. Programs were first written in ILPL and then translated into CIF for verification purposes. The intent was then that an implementation in a real language would somehow be generated form the ILPL, or perhaps ILPL would be compiled directly. However, no ILPL compiler was ever written for any machine. With the use of the Modula language for KSOS, ILPL is now considered obsolete--Modula is currently the only implementation language supported by tools for automatic translation into CIF. With this direct translation of real programs into CIF, there is now no need for the notion of "abstract implementation" in ILPL. ILPL is mentioned here for historical purposes and because it is extensively discussed in the HDM handbook [LEVI79].

Note that the only reason for separating the CIF implementation from the real implementation is practical: the verification tools need only interface to one common implementation language (CIF) for different target languages. It is presumably easier to write CIF translators for different languages than to modify the verification tools, though this has not yet been done extensively.

The entire software development process under HDM need not necessarily proceed in a "pure" top-down manner as described, even though the result is always a hierarchical decomposition of the system into modules of specifications and programs. It is also not necessary for the specification and implementation to be fully completed before any verification begins. It is desirable for verification to take place on an incremental basis so that flaws in the design, detected by verification, can be corrected before the system is completed. Because reverification of an entire system is costly in terms of computer time, SRI has just begun building tools to allow only affected portions of a system to be reverified when a

change is made. Of course initial verification of a system is extremely costly in both manpower and computer time. A second verification of the same system takes little additional manual effort.

4.3.3.2 The Ina Jo Methodology

Ina Joe is the name System Development Corporation applies to its high-order non-procedural specification language and also until recently to the overall hierarchical design and verification methodology which is based upon the Ina Jo language. This methodology is now known as the Formal Development Methodology (FDM). The complete methodology provides features which are intended to enforce the rigorous development and subsequent verification of computer systems. These features include:

- Identification and specification of relevant requirements;
- Design specifications;
- Verification of specifications with respect to requirements;
- Program design specifications; and
- Verification of program implementation.

Among the several automated tools which are provided to support the methodology, by far the most visible is the Ina Jo language. Also provided is a syntax checker/theorem generator (the Ina Jo processor), an interactive theorem prover (ITP), and a set of verification condition generators (VCGs). The choice of implementation language can be made subsequent to development of the specifications, but each new implementation language requires that a new VCG be written. A subset of PASCAL has been used for implementation in several demonstration applications and at one point SDC was developing its own implementation language, FREGE.

Figure 4-4 gives a highly idealized illustration of the interaction of the basic elements of the methodology. Upward arrows on the right indicate hierarchical correspondences (which must be verified using the ITP) defined by the mappings specified on Levels 1, 2, ..., n-1, and downward arrows on the left indicate implied translations of the top level correctness criteria to lower specification levels. The Level 1 Specification is mapped downward to the program code, and both are used to generate verification conditions. Formal verification of correctness criteria against the corresponding level specification is

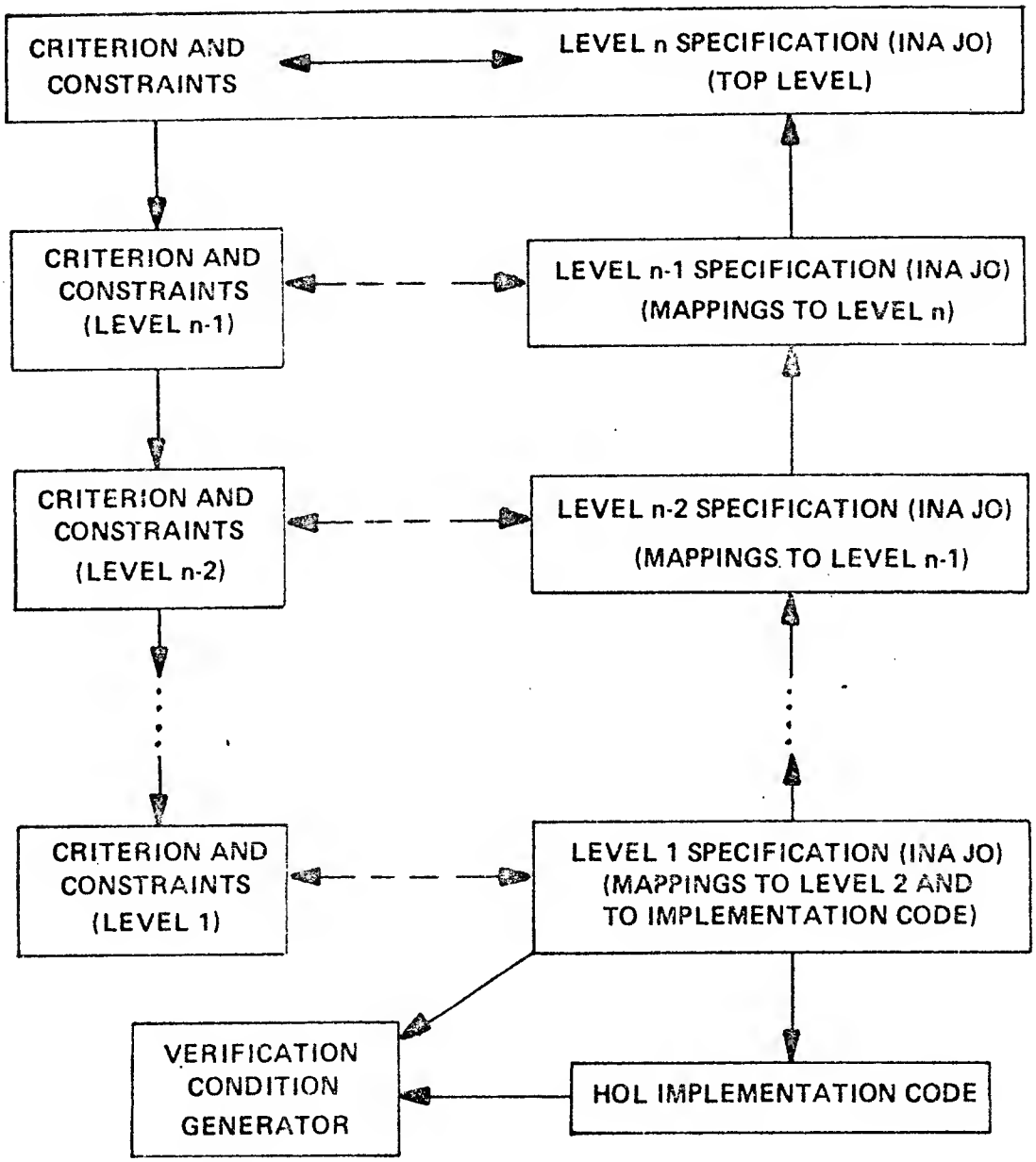

FIGURE 4-4. COMPONENTS FOR THE INA JO METHODOLOGY

carried out on the top level only (since the verification on lower levels is implicit by transitivity). These elements and their interaction will be discussed in detail in the following sections.

4.3.3.3 The Gypsy Verification Environment

The Gypsy Verification Environment is being developed as part of the Certifiable Minicomputer Project (CMP) at the University of Texas under Donald I. Good. The development of the methodology upon which Gypsy is based began in September, 1974, motivated by a need to build small-scale systems that perform critical functions with very high reliability. The objective of the methodology was to build formally verified software that runs on fail-secure hardware [GOOD78a]. One of the specific goals has been the development of a formally verified communications processing system of 1000-2000 lines of code. Communications processing was selected as the particular applications area because these applications typically are small-scale systems that contain many problems common to more general systems. A specific application area also was desired in order to keep the development of the methodology clearly directed at solutions to the problems of real systems. By systems standards, systems of 1000-2000 lines are small; but by formal verification standards, they are very large. Verifications of this scale required that the methodology be developed with equally strong emphasis on both the theoretical and the practical problems of verification.

The Gypsy methodology is based on a wide range of structured programming, formal proof, and specification methods. Throughout the development, the emphasis has been on integration: horizontal integration of programming, specification, and proof methods; and vertical integration of methods, languages, and tools. This integration has resulted in the development of a language, also referred to as Gypsy, that is both a formal specification language and a verifiable high-level language for systems programming. Together with the verification methods, the language allows specification, implementation, and verification to proceed incrementally and in parallel. The methodology supports automated proofs involving concurrency, exception handling, and data abstraction with access control.

The emphasis in the Gypsy Verification Environment has been on implementation proofs rather than design proofs. Verifiability is a major goal of the language design, and the verification system is designed for proving the correspondence between specification and implementation. However, it is also possible to use the verification environment to prove properties of Gypsy specifications.

The Gypsy Verification Environment is intended to be a complete verification system. It includes a syntax-directed editor, a parser, a verification condition generator, a theorem prover, an executive, and a separate compiler. Figure 4-5 shows the relationships between these components. The syntax-directed editor allows the user to directly compose parsable Gypsy statements. The parser checks Gypsy specification and implementation statements for both syntactic and semantic errors, and produces an intermediate form of code for use by the rest of the system. The verification condition generator determines all the paths through a procedure or function, and generates a verification condition (theorem) for each path. The theorem prover attempts to prove each theorem, with assistance from the user as necessary. The executive coordinates the other components, suggests possible actions to the user, and handles terminal communication. There is also a separate cross-compiler which accepts output from the verification system and produces code for the LSI-11.

A Gypsy interpreter for use in program testing is currently under development. MITRE has developed a tool for security flow analysis for use on a limited subset of Gypsy specifications.

4.3.3.4 The Affirm System

AFFIRM is an interactive system for program specification and verification, developed under the Program Verification Project at the USC Information Sciences Institute. It is an experimental system intended to test the application of current research ideas to nontrivial programs.

The description which follows is derived from the system documentation [AFFI79], other published material, and some experience using the system on a security verification example [MILL79]. That example, a substantial one, was specified and verified successfully in two days.

The abstract data type approach to program development regards the top level description of the system as a program, or collection of programs, which are "abstract" in the sense that they perform operations on data structures like queues or sets that are not primitive, i.e., not provided automatically by the programming language. Instead, such data structures have to be manipulated and accessed by a set of operations implemented as subroutines. In AFFIRM, an abstract data type specification is a set of "algebraic axioms" or properties of the operations. The axioms can be used to prove any assertions about the usage of the data type, without knowing how the operations are implemented. Of the systems investigated, only AFFIRM explicitly features the use of algebraic axioms for data

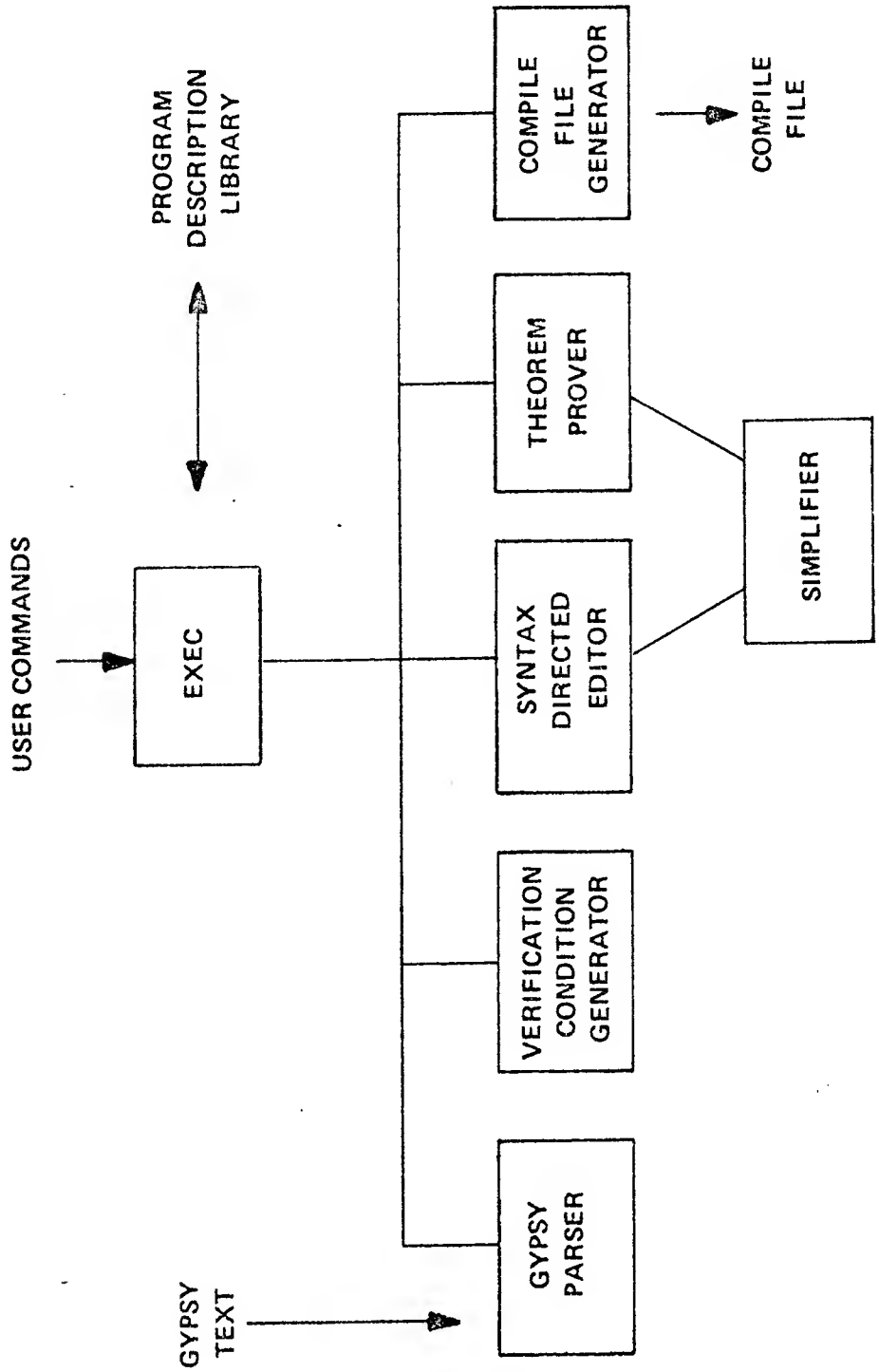

FIGURE 4-5 GYPSY PROGRAM DESIGN SYSTEM

type specification, although similar axioms play a role in the Stanford Verifier, and they may be included as "lemmas" in Gypsy.

The AFFIRM concept of hierarchical development involves manual recoding of higher level programs into different languages as well as the refinement of data structures. The top level programs are LISP-like recursive functions; these are implemented abstractly in a variant of Pascal; and the concrete implementation could be in a third language not directly supported by AFFIRM, such as Bliss, used in the Delta Experiment [GERH79].

The system has a database component that provides facilities for insertion, deletion, and retrieval of specifications, programs, and proofs [MUSS79]. Already in the data base is a library of specifications for data structures such as sets, sequences, and stacks, which are commonly used in the implementation of higher level structures.

The AFFIRM theorem prover may be used effectively at all levels. At the top level, it can be used to prove properties suggested by the user. Properties provable at the top level describe the cumulative effects of sequences of function calls. Access control properties and possibly more general statements about storage channels could be stated this way. Invariants, such as various forms of the *-property, can be proved by "data type induction". Between each successive pair of levels, one can prove the correctness of the implementation.

4.3.3.5 The Stanford Pascal Verifier

The Stanford Verifier is a system for program verification. It is a prototype system currently under development by D. C. Luckham and others at the Stanford Artificial Intelligence Laboratory.

The system consists primarily of a verification condition generator for a version of Pascal, called Pascal Plus, and a theorem prover. There is no explicit support for requirements definition, high-level specifications, abstract data types, or hierarchical program development.

The theorem prover is not interactive, but it runs very quickly. The user does not have to suggest substitutions, subgoals, or case hypotheses. Knowledge about the problem domain which is likely to be of use to the theorem prover is prepared ahead of time in a "rule file". If the theorem prover fails to prove a true verification condition the user's only recourse is to improve the rule file.

4.3.4 R&D Plan

This section presents the essence of a high-level technical plan for the Department of Defense Trusted System Verification Technology R&D Program. The plan covers the requirement, major objectives, technical background and present capability, technical approach, and risks for the Trusted System Verification Technology R&D Program. It covers an eight-year R&D program beginning in FY80. The plan was prepared for the Initiative by Ann Marmor-Squires at NSA [MARM80].

4.3.4.1 Requirement

The use of computers within DoD and the Intelligence Community as significant components of critical systems has increased rapidly over the last decade. These systems, which collect, analyze, store and retrieve classified data, and perform highly critical control functions, demand correct functioning of software. In developing software that must satisfy such rigorous security requirements, the ability to convincingly demonstrate that the software is performing its specified function correctly and that the integrity of the data is maintained, is of vital concern. Recent advances in programming methodology and program verification technology have significantly improved the state of the art in software development and maintenance. However, these efforts are largely experimental and further R&D is needed to achieve the capability of supporting the systematic development of verified and trusted software.

4.3.4.2 Objectives

The ASD(C3I) Trusted System Verification Technology R&D Program is intended to support the goals of the Initiative. The objective of the Program is to develop for DoD the technical capability for the effective practical verification of trusted ADP systems. Achieving this capability will fill a major gap that exists in providing complete technical solutions to the multilevel security problem, will provide a technically sound basis for the trusted system approval process within DoD, and will assist computer manufacturers and DoD contractors in developing trusted and verified systems.

Although the state-of-the-art in verification technology has improved over the last decade, further R&D is needed to make this technology practical and effective. Because of the lack of a complete verification technology capability, the near-term prototype trusted computer systems currently under development (KSOS-6, KSOS-11, KVM/370) will not be fully

verified; this is a critical element missing from a convincing demonstration to the computer industry. The Trusted System Verification Technology R&D Program will provide the needed technology development and demonstrations of its practical application to the development of trusted systems.

As noted in earlier sections, the development of an effective mechanism for approving trusted ADP systems for DoD use is a major goal of the Computer Security Initiative. This must be accomplished within the framework of the existing approval procedures. Current DoD policy permits the use of ADP systems in multilevel secure applications when the Designated Approving Authority for a particular DoD site is convinced that the overall security environment, including external, physical and administrative security measures, is sufficient to overcome the risks against the system. The task of the approving authority is an exceedingly complex one, especially with the advent of software and hardware systems concurrently processing information for multiple users at different access levels and at diverse locations. This task is further complicated when computer systems are required to communicate and share data, as is becoming commonplace within DoD. In addition to reviewing personnel and administrative security, TEMPEST requirements, site and application requirements, the approval authority must determine that the measures taken in the system design and implementation are sufficient to permit use of the system in a multilevel secure mode. A formal verification process is necessary to provide sound technical advice to the approving authority for effective decision making. This judgment needs to be based upon an evaluation of the software methodology used in the development and verification of the system under consideration for approval. The software development and verification methodology must be a rigorous process in order to insure the integrity of the system design and implementation.

The R&D Program described in this plan will help provide the necessary technology to achieve an effective approval process for trusted systems. The approval process must be efficient, consistent and enable formal assessments based upon accepted criteria. In addition to providing the DoD with an effective and practical verification technology, evaluation guidelines and criteria will be developed under this Program for formally assessing software development methodologies and automated support systems. Used in conjunction with technical appraisals of the environment and applications for which a particular system is suitable, this will assure a reasonable approval process for use in trusted systems in widespread application areas.

Involvement of the computer manufacturers in developing trusted ADP systems is the third objective of the Initiative. is not in a position to develop, support and maintain their own computer systems, except for some special-purpose applications. Encouraging computer manufacturers to develop new computer systems that will be suitable for use by DoD in security applications is essential for the widespread availability of trusted systems. Assuring that the system developed meets its specifications is a necessary component of assuring that the system can be trusted. The Program described herein will not only provide for DoD an effective practical software verification technology, but will make it available to the computer manufacturers and DoD contractors for their use in developing high integrity ADP systems for DoD use.

4.3.4.3 Technical Background and Present Capability

The widespread recognition of the "software problem" resulted in much research attention focusing on the development of highly reliable quality software. Economic pressures to reduce the high cost of developing and maintaining software coupled with numerous R&D efforts in programming methodology have improved the state-of-the-art in the short term. Within the last five years, much of the programming methodology research has focused on formalizing the development of software into a systematic and rigorous process having a firm mathematical foundation. Methodologies which make the development process precise facilitate the formal verification of both the design and the implementation. Even if the verification is never carried out, the discipline involved in using these methodologies has already served a useful purpose by aiding in the elimination of many of the difficulties of program development. This research, although not particularly oriented towards the development of trusted software, has been successfully applied in various degrees to several prototype trusted systems under current development (KSOS-6, PSOS, KVM/370, KSOS-11). These efforts are all supported by various components of DoD.

Automated tools to adequately support these new methodologies are crucial to their successful widespread practical application outside of the R&D laboratory. Unfortunately, the development of the tools has lagged behind the evolution of the methodologies. Increased development activities are required to produce a comprehensive integrated set of tools to support the entire development process and demonstrate its effectiveness on large systems.

Currently, these methodologies and support tools are in the early stages of practical application. There has been

limited experience in using them; much of this experience has been limited to the developers themselves. The applications have primarily been small- to medium-scale systems, many of which have only been carried to the detailed design phase. None of the prototype trusted system developments has been carried out to completion (i.e., a fully verified implementation). Such an effort today would be very costly in terms of time, people and machine resources and would probably not result in an efficient implementation. In addition, although there has been some practical experience with the existing automated support tools on several projects, they are all still to be considered experimental. Further work is needed to produce a comprehensive support system for the development of verified and trusted software.

In summary, to achieve the widespread availability of trusted ADP systems, DoD needs mature and stabilized methodologies for the entire software development and maintenance process for trusted systems. These methodologies must be fully supported by comprehensive integrated development and verification environments using effective computational resources. Their effective application must then be demonstrated to the computer industry.

4.3.4.4 Technical Approach

The Trusted System Verification Technology R&D Program is an eight year effort that will be carried out in three phases. Phase I (FY80 - FY82) is a three-year consolidation phase; Phase II (FY83 - FY85) will be a three-year design and prototype implementation phase; Phase III (FY86 - FY87) will be a two-year refinement and application phase.

The primary objective of the Phase I tasks is to bridge the gap between today's experimental verification efforts and the development of the next generation development and verification environment for use in the late 1980s. Phase II will design and implement a prototype development and verification environment for trusted systems development. Phase III will refine this prototype implementation and demonstrate its effectiveness and transfer the necessary technology widely within DoD and industry for their use in the development and verification of trusted systems in the late 1980s.

The approach that is being taken in Phase I to carry out its objective of bridging the gap is to:

(a) Demonstrate the complete formal specification, implementation and verification of several useful trusted system applications.

(b) Bring several current specification and verification methodologies and their associated languages and support tools to a level of completion sufficient for interim use during this phase;

(c) Perform a comparative analysis of the chosen methodologies focusing on their strengths, weaknesses and limitations;

(d) Continue basic verification technology research seeking break-throughs to make the technology more practical and effective;

(e) Track the Ada language and programming environment efforts for potential integration of the Phase II efforts;

(f) Specify the requirements for the next generation development and verification environment for trusted software development;

(g) Transfer technology to DoD and industry of the current generation completed methodologies and supporting languages and tools.

Work on items (b) and (c) has begun under a contract with Digicomp Research Incorporated. The program, managed by RADC, has three parts. The first part, begun in September 1980, is a survey of three prominent specification and verification methodologies to determine strengths and weaknesses of each and to identify areas which could be enhanced to make the methodologies more useable. During the second part, subcontracts will be let to accomplish some of the enhancement identified in Digicomp's survey. In the third part of the program, Digicomp will apply at least two of the enhanced methodologies to a secure database design in order to evaluate their improved useability.

At the end of Phase I (FY83), we should be in a position to begin the development of the next generation development and verification environment. This development will be carried out during Phase II and will be based on the experience gained in completing the current generation methodologies and in demonstrating their use on several trusted system examples, the verification technology research performed and the initial set of requirements developed for the next generation environment. The approach that will be taken in Phase II will be to:

(a) Refine the major issues that need to be resolved in undertaking the development of the next generation development and verification environment for

trusted systems, e.g., suitability of Ada and/or other languages as core language(s) for which the environment will be developed and means of providing effective computational resources so that the environment may be used in a practical and efficient manner;

(b) Design and specify the next generation development and verification environment for trusted software development;

(c) Implement a prototype of the next generation environment;

(d) Continue technology transfer within DoD components and the computer industry of the current generation environment and begin the introduction of the prototype next generation environment for experimental applications.

At the end of Phase II (FY85), we should be in a position to refine the prototype environment into a widely available "production-quality" system and demonstrate its effectiveness.

The approach that will be taken in Phase III will be to:

(a) Refine the prototype implementation of the next generation development and verification environment into a "production-quality" system;

(b) Demonstrate its practical and effective use on suitably chosen trusted systems applications;

(c) Expand the user community within DoD components and transfer technology of the next generation environment to the computer industry for their use in developing trusted systems.

At the successful completion of Phase III (FY87), we should have achieved the technical capability for the effective and practical development and verification of trusted systems of significant size and complexity. There will be a sizeable user community within DoD familiar with the technology and it will have been demonstrated and transferred to the computer industry for their use in developing trusted systems for DoD.

During all three phases, the most advanced verification technology we have will be made available to the Evaluation Center for use in evaluating system security. Guidelines and criteria will also be developed for the Evaluation Center to determine the effectiveness of computer manufacturers' development methodologies.

4.3.4.5 Risks

The efforts being proposed under the Trusted System Verification Technology R&D Program are exploratory research and advanced development, and as such, involve some risk. They represent considerable expansion over previous work and will incorporate significant advances. The recognition of the "software problem" has resulted in much research focusing on reliable software development. Several methodologies having firm theoretical foundation have emerged for developing formally specified and verified software. Their successful widespread application outside of R&D laboratories requires automated tools to adequately support them; unfortunately, the development of automated support tools has lagged behind.

Several R&D projects under DoD sponsorship have explored some practical issues and have produced experimental implementations. These efforts undoubtedly will serve as a basis for the proposed R&D Program; however, there remain numerous technical issues and tradeoffs and pragmatic considerations that need to be investigated in order to support the systematic development of verified and trusted software of substantial size and complexity. Some of these considerations include: (a) limitations of verification in trusted system development (i.e., "complete" correctness will not be achievable); (b) acceptance of new language(s) for widespread use in trusted system development (if integration with Ada is appropriate and feasible, risk would probably be lessened but other complications may be introduced); (c) effective computational resources need to be developed to realistically carry out the verification process. The potential benefit of the proposed R&D Program applied to the development of trusted systems should outweigh the technical risks involved.

REFERENCES

[ABBO76] Abbott, R. P., et al., "Security Analysis and Enhancements of Laboratory," Livermore, CA, National Bureau of Standards, Washington D.C., NBSIR 76-1041 (April 1976).

[ADAM79] Adams, J. A., "Computer Security Environmental Considerations," IBM Federal Systems Division, Arlington, VA, contract MDA 903-79-C-0311, (August 1979).

[AFFI79] AFFIRM System Documentation, USC Information Sciences Institute, (November 1979): AFFIRM Reference Manual (D. H. Thompson, ed.); AFFIRM Collected Papers; AFFIRM Type Library (S. L. Gerhart, ed.); AFFIRM Annotated Transcript (S. L. Gerhart, ed.); AFFIRM User's Guide (S. Gerhart, ed.);

[AMBL76] Ambler, A., "Report on the Language GYPSY," University of Texas at Austin, ICSCA-CMP1, (August 1976).

[ANDE72] Anderson, J. P., "Computer Security Technology Planning Study," J. P. Anderson and Co., Fort Washington, PA, USAF Electronics Systems Division, Hanscom AFB, MA. ESD-TR-73-51. Vols I and II, (October 1972). AD-758206 and AD-772806.

[BBN77] BBN Report 1822, Appendix H, Bolt, Beranek, and Newman, Cambridge, MA, (1977).

[BELL73] Bell, D. E. and L. J. LaPadula, "Secure Computer Systems," ESD-TR-73-278, Volume I-III, The MITRE Corporation, Bedford, MA, (November 1973 - June 1974).

[BELL74] Bell, D. E. and L. J. LaPadula, "Secure Computer Systems: Mathematical Foundations and Model," M74-224, The MITRE Corporation, Bedford, MA, (October 1974).

[BERS79] Berson, J. A. and G. L. Barksdale, Jr., "KSOS--Development Methodology for a Secure Operating System," Proceedings of the 1979 National Computer Conference, (June 1979), pp. 365-371.

[BIBA75] Biba, K. J., "Integrity Considerations for Secure Computer Systems," ESD-TR-76-372, The MITRE Corporation, Bedford, MA, (June 1975).

[BONN80a] Bonneau, C. H., "Secure Communications Processor Kernel Software (Part I)," CPG8656A1, Honeywell Incorporated, Avionics Division, St. Petersburg, FL, (February 1980).

[BONN80b] Bonneau, C. H., D. B. Cameron, and D. S. Lane, "Secure Communications Processor Kernel Software (Part II)," CPG8656A1, Honeywell Incorporated, Avionics Division, St. Petersburg, FL, (February 1980).

[CARL75] Carlstedt, J., R. Bisbey, and G. Popek, "Pattern Directed Protection Evaluation," USC Information Sciences Institute, ISI/75-31, (January 1975).

[CHEH80] Cheheyl, M. H., M. Gasser, G. A. Huff, and J. K. Millen, "Secure System Specification and Verification: Survey of Methodologies," MTR-3904, The MITRE Corporation, Bedford, MA, (February 1980).

[DEWO79] DeWolf, J. B. and P. A. Szulewski, "Summer Study on Air Force Computer Security (1979)," Charles Stark Draper Lab, Cambridge, MA, (October 1979), AFOSR-TR-80-0094, AD-B043742L.

[FEIE77] Feiertag, R. J., et. al., "Providing Multilevel Security of a System Design," Proceedings ACM Sixth Symposium on Operating System Principles, (November 1977).

[FURT78] Furtek, F. C., "A Validation Technique for Computer Security Based on the Theory of Constraints," ESD-TR-78-182, The MITRE Corporation, Bedford, MA, (December 1978).

[GERH79] Gerhart, S. L. and D. S. Wile, "Preliminary Report on the Delta Experiment," Specifications of Reliable Software, IEEE Catalog No. 79 CH 1401-9C, (April 1979), pp. 198-211.

[GOLD79] Gold, B. D., et al., "A Security Retrofit of VM370," Proceedings of the 1979 National Computer Conference, (June 1979), pp. 335-344.

[GOOD78a] Good, D. I., R. M. Cohen, and L. W. Hunter, A Report on the Development of Gypsy, ICSCA-CMP-13, The University of Texas at Austin, (October 1978).

[GOOD78b] Good, D. I., R. M. Cohen, C. G. Hoch, L. W. Hunter, D. F. Hare, Report on the Language Gypsy: Version 2.0, ICSCA-CMP-10, The University of Texas at Austin, (September 1978).

[HINK75] Hinke, T. H., and M. Schaefer, "Secure Data Management System," RADC-TR-75-266, System Development Corporation, Santa Monica, CA, (November 1975).

[HOLT78] Holt, R. C., et al., "The EUCLID Language: A Progress Report," Proceedings of ACM 78 Conference.

[KIRK77] Kirkby, G., and M. Grohn, "On Specifying the Functional Design for a Protected DMS Tool," ESD-TR-77-140, I. P. Sharp Associates, Ltd., Ottowa, Canada, (March 1977).

[KSOS78] "Computer Program Development Specifications (Type B-5) - Secure Minicomputer Operating System (KSOS)," Ford Aerospace and Communications Corporation, Western Development Laboratories Division, Palo Alto, CA, (September 1978).

[LAMP73] Lampson, B., "A Note on the Confinement Problem," Communications of the ACM, 10, (17 October 1973), pp. 613-615.

[LEVI79] Levitt, K. N., L. Robinson, B. A. Silverberg, The HDM Handbook, Vol. I-III, Menlo Park, CA, Computer Science Laboratory, SRI International, (June 1979).

[LIPN75] Lipner, S., "A Comment on the Confinement Problem," Fifth Symposium on Operating System Principles, Austin TX, (November 1975).

[MARM80] Marmor-Squires, A. B., "Trusted System Verification Technology R&D Program," National Security Agency, CSRS-TN-8004, (July, 1980).

[MCCA79] McCauley, E. J., P. J. Drongowski, "KSOS--The Design of a Secure Operating System," AFIPS Conference Proceedings, Vol. 48, (June 1979), pp. 345-353.

[MILL76] Millen, J., "Security Kernel Validation in Practice," Communications of the ACM, Vol 19, No. 5, (May 1976).

[MILL79] Millen, J. K., "Operating System Security Verification," The MITRE Corporation, M79-233, (September 1979).

[MUSS79] Musser, D. R., "Abstract Data Type Specification in the AFFIRM System," Specifications of Reliable Software, IEEE Catalog No. 79 CH 1401-9C, (April 1979), pp. 47-57.

[NIBA79a] Nibaldi, G. H., "Specification of a Trusted Computing Base (TCB)," M79-228, The MITRE Corporation, Bedford, MA, (November 1979).

[NIBA79b] Nibaldi, G. H., "Proposed Technical Evaluation Criteria for Trusted Computer Systems," M79-225, The MITRE Corporation, Bedford, MA, (October 1979).

[POPE78b] Popek, G., and D. Farber, "A Model for Verification of Data Security in Operating Systems," Communications of the ACM, (September 1978).

[POPE79] Popek, G. J., et al., "UCLA Secure UNIX," Proceedings of the 1979 National Computer Conference, (June 1979), pp. 355-364.

[ROUB77] Roubine, O. and L. Robinson, Special Reference Manual, SRI International, Menlo Park, CA, (January 1977).

[SALT75] Saltzer, J. H., "The Protection of Information in Computer Systems," Proceedings of the IEEE, Vol. 63, No. 9, (September 1975).

[SCHR77] Schroeder, M., D. Clark, and J. H. Saltzer, "The MULTICS Kernel Design," Proceedings of the Sixth Symposium on Operating Systems Principles, West Lafayette, Indiana, (November 1977).

[SMIT75] Smith, L., "Architectures for Secure Computing Systems," ESD-TR-75-51, The MITRE Corporation, Bedford, MA, (April 1975).

[TANG78] Tangney, J. D., "Minicomputer Architectures for Effective Security Kernel Implementations," ESD-TR-78-179, The MITRE Corporation, Bedford, MA, (October 1978).

[TANG80] Tangney, J. D., "History of Protection in Operating Systems," MTR-3999, The MITRE Corporation, Bedford, MA, (July 1980).

[TROT80] Trotter, E. T. and P. S. Tasker, "Industry Trusted Computer System Evaluation Process," MTR-3931, The MITRE Corporation, Bedford, MA, (May 1980)

[WALK79] Walker, G., R. Kemmerer, and G. Popek, "Specification and Verification of the UCLA UNIX Security Kernel," Proceedings of ACM SIGOPS Conference, (December 1979), to be published in Communications of the ACM.

[WARE70] Ware, W. H., Ed. "Security Controls for Computer Systems," R-609-1, Rand Corporation, Santa Monica, CA, (February 1970, reissued October 1979).

[WOOD77] Woodward, J. P. L., and G. H. Nibaldi, "A Kernel-Based Secure UNIX Design," MTR-3499, The MITRE Corporation, Bedford, MA, (November 1977).

[WOOD79] Woodward, J. P. L., "Applications of Multilevel Secure Operating Systems," Proceedings of the 1979 National Computer Conference, (June 1979), pp. 319-328.

ATT. H

KEYNOTE ADDRESS

COMPUTER SECURITY INITIATIVE

August 10, 1981

It is a pleasure to welcome you to this Seminar and to speak briefly with you about computer security, the recent developments within the Department of Defense and the Intelligence Community and the challenges that lie ahead.

As Dr. Gerald P. Dinneen, former Assistant Secretary of Defense for C^3I defined at the first of these Seminars two years ago, a "trusted" computer system is one with sufficient hardware and software integrity to allow its use for the simultaneous processing of multiple levels of classified or sensitive information.

The need for trusted computer systems is very real and growing rapidly. Factors influencing this need are:

- the growing use of automated information handling systems throughout the DoD and the Intelligence Community and in particular the linking of these systems into major networks;
- increasing requirements for controlling access to compartmented and sensitive information;
- the requirement for broader dissemination of information both within and beyond the community;
- growing difficulties with obtaining required numbers of cleared personnel, both military and civilian.

Despite continuing internal efforts to develop special purpose trusted systems for unique needs, we already rely very heavily on the products of the computer industry to meet our information processing requirements, and this

dependence will continue to grow significantly in the future. It is therefore very gratifying to observe the progress being made by the computer industry in applying computer security technology as represented by the industry presentations at this and the previous Seminars.

It is very important, also, that the Department of Defense and the Intelligence Community develop sufficient expertise to be able to evaluate the integrity of computer software and systems developed by industry and government, and that we be able to determine suitable physical and administrative environments for their application. We have had scattered efforts over the past several years to evaluate specfic systems for specific installations. But these efforts have always been more or less ad hoc, and because of the extensive technical background required, expensive to carry out.

I am very pleased therefore to announce today the establishment of a Computer Security Technical Evaluation Center for the Department of Defense and the Intelligence Community at the National Security Agency. Last fall, as Director of NSA, I enthusiastically endorsed the establishment of this Center at NSA as a new and separate function. I am very pleased with the progress being made in setting up the Center and I remain strongly committed to its success.

I would like to make several observations about the Center and some of its relationships:

- Because the private sector computer manufacturing community is the primary source of ADP systems, the Center's role will be to work with the manufacturers, deriving as much system integrity as possible from industry developed systems. This is a rather sharp contrast to the NSA's more traditional communications security role where the government has the dominant technical role.

- The Center will have a difficult task developing procedures which assure protection of sensitive portions of a system which the government does not own. Simply classifying security related portions of a system built by industry won't work since the government represents such a small portion of the overall market that the manufacturers may well decide not to sell to the government rather than accepting the limitations imposed by classification. This, in the end, might lead to a highly undesirable situation where private sector users (e.g., banks, insurance companies) have higher integrity systems than the government.
- But sensitive portions of systems and the known vulnerabilities that remain must be protected, in the interests of both the government and the manufacturers. It is quite likely therefore that the most sensitive portions of the government's analyses will be both classified and proprietary to the manufacturer. Careful, reasoned interaction between the government and industry will be needed to work out suitable working relationships.
- The Center will act in the interests and for the benefit of the Department of Defense and the Intelligence Community. Its evaluation will not be intended for use by other than the DoD. It will not make general product endorsements. But as with the Qualified Products List procedures (as prescribed in the DoD Defense Acquisition Regulations), the relative merit of a system in the hierarchy of evaluated products may be available publicly in order to provide incentive and encouragement for manufacturers to develop trusted systems and private sector users to employ them.

Agencies are being encouraged to establish or enhance their own technical security test and evaluation capabilities to ensure widespread use and availability of trusted computer systems. The computer manufacturing community must work closely with the Center and these Service organizations to ensure that reasonable products are available for use in sensitive applications.

In conclusion, I would like to restate my awareness of the importance of this problem area, my enthusiasm for the establishment of the Evaluation Center, and my deep and continuing interest in its success. I encourage you to participate fully in this Seminar, ask the tough questions, learn all you can, and then go out and apply what you have learned so that we may all have trustworthy computers in the very near future.

ATT. I

ADDRESS BY LTG LINCOLN D. FAURER, DIRECTOR NSA AT IEEE COMPUTER CONFERENCE 81 WASHINGTON, D.C. 15 SEPTEMBER 1981

DOD COMPUTER SECURITY -- A NEW INITIATIVE